MASTER EN SCIENCES INFORMATIQUES

Une extension de TOGAF orientée Domain-Driven Design pour la gouvernance d'architectures de SI

Clément, David-Yvon

Award date:
2017

Awarding institution:
Universite de Namur

UNIVERSITÉ DE NAMUR
Faculté d'informatique
Année académique 2016-2017

Une extension de TOGAF orientée Domain-Driven Design pour la gouvernance d'architectures de SI

David-Yvon CLEMENT

Promoteur : _____ (Signature pour approbation du dépôt- REE art. 40)
Michaël PETIT

Mémoire présenté en vue de l'obtention du grade de Master en Sciences Informatiques.

Résumé

Il existe, actuellement, différents cadres d'architecture d'entreprise dont notamment TOGAF, l'un des plus connu à ce jour et possédant la plus grande communauté de personnes certifiées. L'Open Group, responsable de TOGAF, prévoit plusieurs extensions à ce dernier afin de permettre la considération d'aspects spécifiques ou de supporter la mise en œuvre de différents styles d'architecture au sein de la démarche d'architecture d'entreprise.

Malgré cela, il existe de nombreux styles d'architectures intéressants et consacrés à la conception logicielle pour lesquels TOGAF ne prévoit pas encore, à ce jour, d'extension.

La conception dirigée par le domaine est un exemple de style d'architecture qui n'est pas encore intégré par TOGAF. Ce style d'architecture est destiné à faciliter la gestion de domaines métier complexes. Ceci en proposant un ensemble de techniques et de recommandations dédicacées à leur gestion, leur compréhension, et qui positionnent les domaines au centre des préoccupations de la conception logicielle.

Le but du présent mémoire est, dans une première phase, de faire le point sur ce qu'est l'architecture d'entreprise, ce qu'est TOGAF et expliquer le style d'architecture DDD.

Dans une seconde phase, nous tenterons de trouver les liens entre TOGAF et DDD en examinant la capacité de TOGAF à intégrer ce style architectural.

Si des adaptations du cadre TOGAF sont nécessaires, celles-ci seront décrites dans ce mémoire. L'évaluation de la capacité qu'à TOGAF à supporter DDD sera également critiquée.

Mots clés

Architecture d'Entreprise, TOGAF, Conception Dirigée par le Domaine, DDD, Style d'Architecture, Open Group, Eric Evans.

Abstract

Nowadays, there are different architecture frameworks and among them is TOGAF, currently one of the most famous and grouping the largest community of certified people. Open Group, which is responsible for TOGAF, is providing several extensions in order to allow the consideration of the company specifications or to support the achievement of different styles during the company architecture steps.

Despite this, there are a lot of interesting architecture styles that are dedicated to the software design for which TOGAF has not done any extensions for the moment. The concept directed by the Domain-Driven-Design is an example of architecture style that is still not integrated by TOGAF. This one is designated to make an easier management of complex business domain, by providing a set of technics and recommendations helping their execution and understanding, and that are positioning domains in the center of concerns and software design.

The aim of this thesis is, in the first place, to focus on what is the company architecture, what is TOGAF, and explain DDD architecture style.

Secondly, we will try to single out the links between TOGAF and DDD, by studying TOGAF's ability to include this architecture style.

If TOGAF's framework adaptations are needed, they will be described in this thesis. TOGAF's ability level to support DDD will be as well criticized.

Keywords

Enterprise Architecture, TOGAF, Domain-Driven Design, DDD, Architecture Style, Open Group, Eric Evans.

Avant-propos

Ce mémoire est le résultat de plusieurs mois de travail et rentre dans le cadre de l'obtention du Master 60 en Sciences Informatiques à horaire décalé.

Ayant déjà une expérience dans le domaine de l'architecture informatique, qui, je dois le dire, me passionne, j'ai souhaité orienter mon travail vers ce domaine de recherche.

Je souhaitais proposer un mémoire portant sur TOGAF, cadre d'architecture d'entreprise publié par l'Open Group, qui définit un ensemble méthodologique complet, à usage libre, basé sur des modèles génériques et des bonnes pratiques, afin de piloter la transformation de l'entreprise et de professionnaliser la fonction d'architecture d'entreprise. Le champ de recherche autour de ce domaine est bien évidemment gigantesque. C'est suite à un défi professionnel, dans le contexte de ma participation à la mise en œuvre d'une démarche d'architecture d'entreprise, que j'ai décidé de me concentrer sur l'évaluation de la capacité d'intégration de l'approche DDD au sein de la démarche TOGAF.

Après quelques discussions avec Michaël PETIT, nous avons pu valider l'intérêt de ce sujet de mémoire. Je le remercie d'ailleurs pour son encadrement et les précieux conseils qu'il m'a apporté tout au long de ce travail.

Le travail présenté dans ce mémoire n'aurait jamais pu se faire sans le support et le soutien d'autres intervenants. Je tiens donc à remercier ceux qui y ont contribué, de près ou de loin.

Merci à la société Mielabelo, de m'avoir permis, au travers de mes diverses expériences, de trouver ma voie dans le domaine de l'architecture logicielle, ce qui a indirectement aiguillé le sujet du présent mémoire.

Merci à Christophe D'Agostino, collègue et ami, sans qui tout cela n'aurait pas été possible. Nous avons décidé, en 2014, de nous lancer ensemble dans cette aventure de reprise des cours à horaire décalé.

Merci à Fabian Dermine, chef du service informatique au SPW-DGO3, pour son soutien, sa confiance et son suivi.

Merci à mes collègues, qui m'ont encouragé tout au long de mon parcours. Tout particulièrement, à Laurent Boucher et Patrick Vandevoorden pour leurs conseils.

Merci à mes parents qui m'ont encouragé tout au long de mes études et d'avoir fait de moi une grande partie de ce que je suis devenu.

Enfin, sans doute l'hommage qui me tient le plus à cœur, je tiens à remercier ma compagne Emilie Lempereur et ma fille Sarah, grâce à leur soutien et aux sacrifices concédés ces trois dernières années, elles m'ont permis de vivre l'aventure, à la fois passionnante et éprouvante, que sont les études en horaire décalé.

Table des matières

Résumé 2
Abstract 3
Avant-propos 4
Table des matières 5
Table des illustrations 6
Abréviations 7
Introduction 8
Partie 1 : Etat de l'art 10
1.1 AE - Architecture d'Entreprise 11
1.1.1 Définition de l'architecture d'entreprise 12
1.1.2 Historique de l'architecture d'entreprise 22
1.1.3 Les domaines de l'architecture d'entreprise 25
1.1.4 Classification des cadres d'Architecture d'Entreprise 28
1.2 TOGAF – The Open Group Architecture Framework 30
1.2.1 Présentation 31
1.2.2 Historique 32
1.2.3 Le contenu de TOGAF 34
1.2.4 TOGAF et les styles d'architecture logiciels 53
1.3 DDD – Domaine Driven Design 54
1.3.1 Présentation 55
1.3.2 Historique 57
1.3.3 Principes 58
Partie 2 : Intégration du style d'architecture DDD au sein de TOGAF 76
2.1 Exécution de la phase préliminaire – Définition de la Capacité d'Architecture 77
2.1.1 Etape : Identifier et établir les principes d'architecture 78
2.1.2 Etape : Sélectionner et adapter un ou des cadre(s) d'architecture 81
2.1.3 Etape : Alimenter le référentiel d'architecture 110
Conclusion 112
Références 114
Bibliographie 114
Webographie 115
Annexes 116
Annexe 1 – Historique détaillé de l'architecture d'entreprise 117
Annexe 2 – L'architecture d'entreprise et les instruments de gouvernance 121
Annexe 3 – Outils d'architecture d'entreprise 125
Annexe 4 – Vue d'ensemble de DDD (détaillée) 126
Annexe 5 – Définitions des éléments de base du métamodèle TOGAF 127
Annexe 6 – Définitions des éléments d'extension du métamodèle TOGAF 129
Annexe 7 – Contexte métier de mise en œuvre 131

Table des illustrations

Figure 1 - ISO/IEC/IEEE 42010 – Métamodèle [ISO/IEC/IEEE42010, 2011] 14
Figure 2 : Ligne du temps de l'AE 24
Figure 3 : Couches de l'architecture d'entreprise [NILES, 2006] 25
Figure 4 : TOGAF – Ligne du temps [ArchiMetric] 33
Figure 5 : TOGAF – Structure générale du document TOGAF [TOGAFGuideDePoche, 2010] 34
Figure 6 : TOGAF – Synoptique du contenu de TOGAF [TOGAF9.1] 35
Figure 7 : TOGAF – Cycle ADM [TOGAF9.1] 37
Figure 8 : TOGAF - Interactions entre Métamodèle, Building Blocks, Diagrammes et Parties prenantes [TOGAF9.1] 43
Figure 9 : TOGAF – Cadre de contenu [TOGAF9.1] 44
Figure 10 : TOGAF – Métamodèle de contenu de base et ses extensions [TOGAF 9.1] 45
Figure 11 : TOGAF – Les constituants d'architecture [TOGAF9.1] 46
Figure 12 : TOGAF – Continuum d'entreprise [TOGAF9.1] 47
Figure 13 : TOGAF – Continuum d'architecture [TOGAF9.1] 48
Figure 14 : TOGAF – Technical Reference Model (TRM) [TOGAF9.1] 49
Figure 15 : TOGAF – Integrated Information Infrastructure Reference Model (III-RM) [TOGAF9.1] 50
Figure 16 : TOGAF – Cadre de Capacité [TOGAF9.1] 51
Figure 17 : DDD – Vue d'ensemble [DDD, 2004] 56
Figure 18 : DDD – Comprendre et communiquer le domaine [DDD, 2004] 58
Figure 19 : DDD – Distillation du domaine métier [DDD, 2004] 60
Figure 20 : DDD – Techniques de préservation de l'intégrité du domaine métier [DDD, 2004] 62
Figure 21 : DDD – Conception Dirigée par le modèle [DDD, 2004] 67
Figure 22 : DDD – Architecture en couches [DDD, 2004] 68
Figure 23 : Niveau d'impact de l'intégration de DDD sur les étapes et la phase préliminaire de TOGAF 77
Figure 24 : TOGAF – Extensions du métamodèle TOGAF [TOGAF9.1] 84
Figure 25 : TOGAF – Domain Driven Design Extensions 85
Figure 26 : TOGAF – Localisation de l'extension DDD dans le métamodèle TOGAF 85
Figure 27 : DDD – Métamodèle des éléments de base de l'extension DDD 87
Figure 28 : TOGAF – Métamodèle détaillé des éléments de base de TOGAF [TOGAF9.1] 92
Figure 29 : Matrice des similarités et nouvelles relations entre éléments de base TOGAF et DDD 95
Figure 30 : Métamodèle des éléments de base de TOGAF et de l'extension DDD 96
Figure 31 : TOGAF – Métamodèle détaillé des éléments de base de TOGAF et de ses extensions [TOGAF9.1] 99
Figure 32 : Matrice des similarités et nouvelles relations entre éléments d'extension TOGAF et DDD 101
Figure 33 : Métamodèle des éléments de base et d'extension de TOGAF et de DDD 102
Figure 34 : Niveau d'impact de l'intégration de DDD sur les artefacts d'architecture TOGAF 105

Abréviations

Abréviation	Signification
SPW	**S**ervice **P**ublic de **W**allonie
DGO3	**D**irection **G**énérale opérationnelle de l'**A**griculture, des **R**essources **N**aturelles et de l'**E**nvironnement
DA	**D**épartement des **A**ides agricoles
DAgri	**D**épartement de l'**Agri**culture
DO	**D**irection de l'**O**ctroi des aides agricoles
OPW	**O**rganisme de **P**aiement **W**allon
PAC	**P**olitique **A**gricole **C**ommune
SIGEC	**S**ystème **I**ntégré de **G**estion **E**t de **C**ontrôle
DEMNA	**D**épartement de l'**E**tude du **M**ilieu **N**aturel et **A**gricole
EWBS	**E**-**W**allonie-**B**ruxelles **S**implification
DTIC	**D**épartement des **T**echnologies de l'**I**nformation et de la **C**ommunication
DFA	**D**irection **F**onctionnelle et d'**A**ppui
SI	**S**ystème d'**I**nformation
IT	**I**nformation **T**echnology
PMO	**P**roject **M**anagement **O**ffice
TOGAF	**T**he **O**pen **G**roup **A**rchitecture **F**ramework
ADM	**A**rchitecture **D**eveloppment **M**ethod
DDD	**D**omain **D**riven **D**esign
TRM	**T**echnical **R**eference **M**odel
III-RM	**I**ntegrated **I**nformation **I**nfrastructure **R**eference **M**odel
SOA	**S**ervice **O**riented **A**rchitecture

Introduction

L'informatique est en évolution constante et les entreprises sont de plus en plus majoritaires à employer ce média dans le but de soutenir et développer leurs activités, et pour ce faire elles ont investi beaucoup de ressources pour le développement de leur système d'information.

Au fil du temps, ces entreprises ont vu leurs parcs applicatifs se remplir avec des applications reflétant les paradigmes et la technologie en vogue, en fonction du moment de leur conception. L'alimentation des Parcs applicatifs est un processus qui s'étale généralement sur la période de vie de l'entreprise. Les parcs applicatifs se sont donc vu peuplés au fil de l'évolution de l'entreprise mais également des technologies. Parfois, des décennies se sont écoulées depuis l'apparition des premières applications, la plupart ont été remplacées, partiellement ou totalement, par d'autres applications. Cette période de transformation continue des entreprises et de leur système d'information a pour résultat d'avoir rendu les parcs applicatifs des entreprises relativement hétérogènes, tant d'un point de vue technologique que pragmatique.

De plus, la plupart du temps, l'absence ou la faiblesse d'éléments permettant de raisonner en accord avec l'écosystème informatique de l'entreprise et des contraintes fortes appliquées sur les ressources (délais, budget) n'ont pas toujours permises une réflexion globale qui aurait guidée les choix de l'entreprise.

Malheureusement à terme, cette hétérogénéité et cette scission entre la vision métier et la vision informatique est devenue une problématique récurrente de ces entreprises. Les coûts de maintenance et la diversité des savoirs nécessaires à la survie de leurs écosystèmes informatiques sont en constante augmentation.

Face à cet état des lieux, une démarche d'architecture d'entreprise est de plus en plus perçue comme salvatrice. Le souhait des entreprises est de faire réapparaitre l'ordre dans le chaos technologie et organisationnel de leurs parcs applicatifs.

Pour aider les entreprises dans cet effort de rationalisation, plusieurs outils existent, dont TOGAF. Ce dernier permet de suivre et de guider les entreprises dans leur démarche de transformation en leurs fournissant un cadre et des règles permettant d'encadrer leur évolution.

Parallèlement à cela, un autre problème existe actuellement dans de nombreuses entreprises : la complexité grandissante de leur domaine métier, qui se retrouve le plus souvent dispersé au travers de leurs applications. Un manque de vision globale et de compréhension de leur domaine métier empêche les entreprises de raisonner de façon valide et exhaustive sur ce dernier. Une des solutions à ce problème est d'accorder une plus grande importance aux domaines métier des entreprises. Il existe de multiples solutions permettant de (re)positionner le domaine métier au centre de la réflexion et des considérations relatives à la conception logicielle. Le style d'architecture DDD est l'un des différents choix possibles pour aider les entreprises dans cette voie.

Le but de ce mémoire est d'évaluer la capacité d'intégration du style d'architecture DDD au sein d'une démarche d'architecture d'entreprise réalisée à l'aide de TOGAF, et de proposer les ajustements nécessaires si ces derniers sont requis.

La complémentarité et le renforcement mutuel de ces deux approches méthodologiques, ainsi que les concepts qui s'en dégagent, ont pour finalité de créer une réelle plus-value dans le cadre d'une mise en œuvre d'une démarche d'architecture d'entreprise, ceci au sein d'une entreprise disposant d'un domaine métier complexe.

Le sujet de ce mémoire vise trois objectifs :

- Comprendre les enjeux de l'architecture d'entreprise, TOGAF et DDD.
- Vérifier la capacité d'adaptation de TOGAF à un style d'architecture particulier.
- Identifier les adaptations nécessaires pour adopter le style d'architecture DDD au sein de TOGAF

La suite du présent mémoire est structurée comme suit :

Partie 1 : Etat de l'art

- 1.1 AE - Architecture d'Entreprise
- 1.2 TOGAF – The Open Group Architecture Framework
- 1.3 DDD – Domaine Driven Design

Partie 2 : Intégration du style d'architecture DDD au sein de TOGAF

- 2.1 Exécution de la phase préliminaire – Définition de la capacité d'architecture

Partie 1 : Etat de l'art

Cette section présente l'état des connaissances générales et réutilisables existantes concernant l'architecture d'entreprise, TOGAF et DDD.

Pour débuter, l'architecture d'entreprise sera présentée de manière générale.

Ensuite nous porterons une attention particulière au cadre d'architecture d'entreprise TOGAF et de la façon dont ce dernier intègre déjà d'autres méthodes de conception logicielle.

Finalement La conception dirigée par le domaine et ses principaux atouts seront détaillés.

La réalisation de cet état de l'art permettra dans la partie suivante de décrire les points de jonctions et les ajustements nécessaires à la réussite de l'intégration de l'approche DDD au sein de TOGAF.

A l'aide d'une bonne connaissance de la situation existante, il sera possible d'identifier ce qui sera jugé comme manquant ou alternatif par rapport aux méthodes et techniques appliquées de manière isolées. Et de cette façon, montrer comment l'association de TOGAF et de DDD permet de tirer profit des deux méthodologies et de créer une synergie forte entre ces deux approches.

1.1 AE - Architecture d'Entreprise

Cette section présente de manière générale l'architecture d'entreprise, ses concepts, ses évolutions et ses manifestations jusqu'à aujourd'hui.

Premièrement, une définition de l'architecture d'entreprise et une présentation de son historique et de ses motivations seront exposées.

Secondement, les méthodes et techniques les plus importantes et utilisées en architecture d'entreprise seront présentées.

Bien entendu, cette description est un instantané et ne peut pas prétendre être exhaustive étant donné que le domaine de l'architecture d'entreprise est en évolution constante et rapide. Cependant, il offre un aperçu des méthodes et techniques actuelles.

1.1.1 Définition de l'architecture d'entreprise

Un bon point de départ pour entamer une présentation de l'architecture d'entreprise est d'en établir une compréhension générale.

Premièrement une décomposition des termes de cette discipline sera réalisée.

Deuxièmement nous allons parcourir des définitions de l'architecture d'entreprise afin d'en prendre connaissance au travers du prisme de différents acteurs clés dans ce domaine.

Finalement, nous tenterons de restituer notre compréhension et notre vision de ce que représente l'architecture d'entreprise.

Architecture

Nous allons maintenant définir ce que l'on entend par « Architecture ». Pour ce faire, nous allons nous baser sur des définitions d'abord générales et ensuite plus spécifiques au domaine de l'informatique, ceci afin d'en ressortir les éléments principaux pour que le lecteur puisse mieux comprendre le concept d'architecture.

L'**architecture**[1], dans ses manifestations les plus courantes, implique la réalisation de plans. Par exemple dans le domaine de la construction : en y précisant tout de la façon dont le toit se connecte aux murs, à la façon dont les tuyaux sont posés, ou encore à l'endroit où les prises électriques sont situées, et ainsi de suite…

Néanmoins, au-delà de son champ d'application dans le domaine de la construction, l'architecture s'applique à des domaines bien plus étendus. L'Oxford English Dictionary nous rappelle qu'il existe deux sens au terme :

« Architecture (Nom)

- *L'art ou la pratique de la conception et de la construction de bâtiments.*
  Le style dans lequel un bâtiment est conçu et construit, notamment en ce qui concerne une période, un lieu ou une culture spécifique.

- *La structure complexe ou soigneusement conçue de quelque chose :*
  - *l'architecture chimique du cerveau humain*
  - *la structure conceptuelle et l'organisation logique d'un ordinateur ou d'un système informatique »* [2]

[OxfordArchitecture]

La notion d'Architecture qui nous occupe dans le cadre de ce mémoire correspond d'avantage au deuxième sens qui lui est donné dans la définition précédente, c.-à-d. lorsqu'elle est appliquée aux systèmes.

Les premières définitions de l'architecture des logiciels et des systèmes se concentrent naturellement sur le caractère structurel de l'architecture, par analogie avec les idées d'architecture dans l'environnement bâti, par exemple :

> *« Une architecture logicielle est un ensemble d'éléments d'architecture (ou, si vous voulez, de conception) qui ont une forme particulière. »* [3]

[Perry&Wolf, 1992]

---

[1] Du latin architectura (« architecture, construction »).

[2] Texte original disponible à l'adresse suivante : https://en.oxforddictionaries.com/definition/architecture

[3] Texte original : Software architecture is a set of architectural (or, if you will, design) elements that have a particular form.

Actuellement, les définitions mettent l'accent sur l'architecture en tant qu'outil d'aide à la décision.

« Une architecture est l'ensemble des décisions importantes concernant l'organisation d'un système logiciel, la sélection des éléments structurels et leurs interfaces à partir desquels le système est composé, ainsi que leur comportement tel que spécifié dans les collaborations entre ces éléments, la composition de ses éléments de structure et de comportement dans des sous-systèmes progressivement plus grands, et le style d'architecture qui guide cette organisation - ces éléments et leurs interfaces, leurs collaborations, et leur composition. » [4]

[Krutchen, 1999]

Notons que toutes les décisions ne font pas partie de l'architecture, seules sont retenues les décisions qui sont fondamentales au système.

Le standard international « ISO/IEC/IEEE 42010 – Systems and software engineering – Architecture description » propose une définition de l'architecture. Ce standard est basé sur un modèle conceptuel des termes et des concepts relatifs à la description de l'architecture. Le modèle est documenté dans la norme à l'aide de diagrammes de classes UML pour représenter les entités et leurs relations (voir Figure 1).

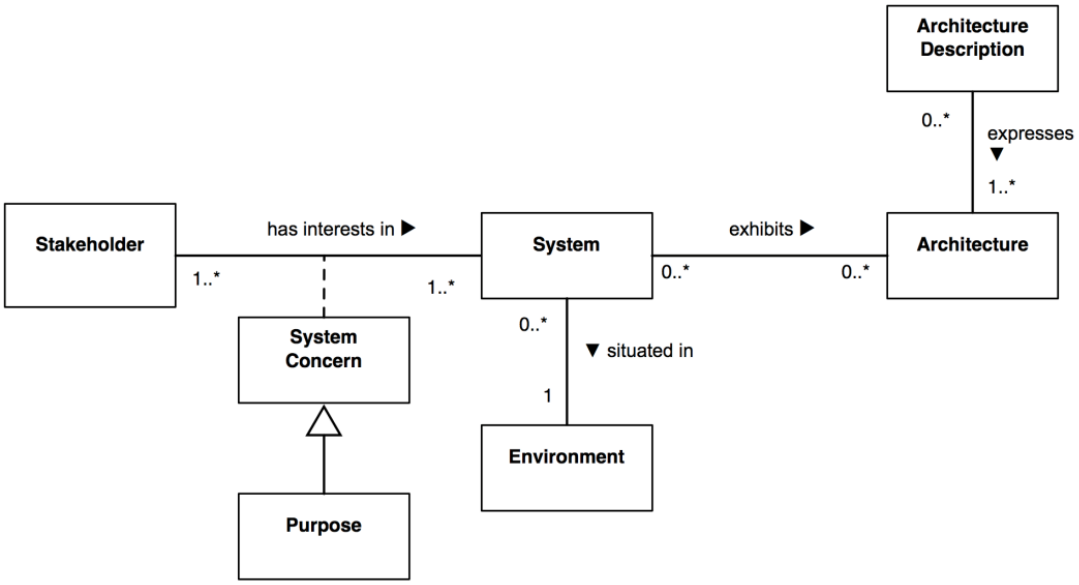

Figure 1 - ISO/IEC/IEEE 42010 – Métamodèle [ISO/IEC/IEEE42010, 2011]

Un système (**System**) est situé dans son environnement (**Environment**). Un environnement peut inclure d'autres systèmes. Les parties prenantes (**Stakeholders**) ont des intérêts dans un système, ces intérêts sont appelés des Préoccupations (**System Concerns**). Les préoccupations peuvent être des objectifs attendus (**Purpose**) du système par les parties prenantes. Les systèmes exposent des

[4] Texte original : "An architecture is the set of significant decisions about the organization of a software system, the selection of the structural elements and their interfaces by which the system is composed, together with their behavior as specified in the collaborations among those elements, the composition of these structural and behavioral elements into progressively larger subsystems, and the architectural style that guides this organization — these elements and their interfaces, their collaborations, and their composition."

architectures (**Architecture**) et une architecture peut concerner plusieurs systèmes. Une description d'architecture (**Architecture Description**) est utilisée pour exprimer une ou plusieurs architectures et une architecture peut faire l'objet de multiples descriptions.

Dans le standard, l'architecture d'un système est définie comme suit :

«Concepts fondamentaux ou propriétés d'un {système} dans son environnement incarné dans ses éléments, ses relations et ses principes de conception et d'évolution».[5]

[ISO/IEC/IEEE42010, 2011]

Ce qui est fondamental pour un système peut prendre plusieurs formes :

- Les éléments qui composent le système.
- Les relations internes et externes du système.
- Les principes de conception et d'évolution du système.

{Système} est une convention ISO (des crochets) pour indiquer que le terme défini se rapporte au domaine des systèmes. Dans le standard le terme système est utilisé comme espace réservé, par exemple il peut se référer à :

- Des systèmes dans l'esprit de la norme ISO 15288 (traduction personnelle) :

  "… qui sont fabriqués par l'homme et peuvent être configurés avec un ou plusieurs des éléments suivants : matériels, logiciels, données, humains, processus, installations, matériaux, entités naturelles … »

- Des produits et services logiciels tels que décrits dans la norme ISO 12207

- Des systèmes à forte intensité de logiciel comme décrit dans le standard ISO/IEC/IEEE42010, ce qui comprend :

  - une entreprise (ou toutes autres agrégations d'intérêts)
  - un système de systèmes
  - une ligne de produits
  - un service
  - un sous-système ou un logiciel
  - …

Dans la définition de l'architecture proposée par le standard ISO 42010, la disjonction, « les concepts ou propriétés » a été choisie pour soutenir deux philosophies différentes qui sont :

[5] Texte original : *Architecture* : fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution

- **L'Architecture comme conception**: une architecture est un concept d'un système dans son esprit.

- **L'Architecture comme perception**: une architecture est une perception des propriétés d'un système.

Sous l'une ou l'autre philosophie, une architecture est toujours abstraite et donc elle n'est pas un artefact. Pour nommer l'architecture lorsqu'elle est exprimée et documentée par des artefacts il convient alors d'utiliser un autre terme, la **description de l'architecture**.

Souvent on utilise (trop) largement le terme «architecture» qui confond l'idée abstraite avec les artefacts capturant cette abstraction, ceci en particulier dans le domaine de l'architecture d'entreprise.

L'architecture d'un système est consciente du système dans son environnement. L'environnement détermine la totalité des influences portant sur le système. Une différence souvent cité entre l'architecture logicielle et la conception logicielle est la suivante : l'architecture logicielle est orientée vers l'extérieur, sur le système dans son environnement, alors que la conception logicielle est orientée vers l'intérieur du système, une fois que les limites sont définies.

Pour conclure notre approche de la définition de l'architecture voici la définition proposée par l'Open Group au travers du document TOGAF 9.1 :

« Architecture :

1. *Une description formalisée d'un système, ou bien, au niveau d'un composant, la description détaillée de ce composant permettant sa mise en œuvre.*

2. *La structure des composants, accompagnées des relations inter-composants, des principes et règles de définition et d'évolution au cours du temps de ceux-ci. »*

[OGF, 2012]

Cette définition ignore la distinction entre l'architecture et la description de l'architecture, ce qui au regard du standard ISO/IEC/IEEE 42010 peut entrainer des confusions dans tout le document TOGAF. On peut en conclure que TOGAF embrasse mais ne respecte pas strictement la terminologie du standard ISO/IEC/IEEE 42010.

Entreprise

Nous allons maintenant définir ce que l'on entend par « Entreprise ». Pour ce faire, nous allons nous baser sur la définition donnée par TOGAF et en ressortir les éléments principaux afin que le lecteur puisse mieux comprendre le concept d'entreprise.

L'**Entreprise**[6] considérée dans le cadre de l'architecture d'entreprise est définie par l'Open Group comme :

"Entreprise :

1. *Typiquement le plus haut niveau de description d'une organisation qui couvre toutes les missions et les fonctions. Une entreprise sera souvent répartie en plusieurs organisations.*

2. *N'importe quel ensemble d'organisations ayant les mêmes buts et/ou un simple objectif identique."*

[OGF, 2012]

Il est intéressant de rappeler que les entreprises ne sont pas uniquement des structures guidées par un intérêt de rentabilité mais peuvent de manière plus large identifier n'importe quel organisme public ou d'intérêt général.

La démarche d'architecture d'entreprise identifie généralement trois aspects importants de l'entreprise :

La stratégie

L'entreprise se dote d'une **stratégie** lui permettant d'atteindre ses objectifs. Cette stratégie est un élément essentiel à considérer attentivement afin de comprendre « le mouvement » de l'entreprise.

Les ressources

L'entreprise à besoins de **ressources** pour fonctionner. L'évaluation des capacités de l'entreprise doit tenir compte des ressources dont l'entreprise dispose pour mener à bien sa stratégie.

L'environnement

L'entreprise évolue au sein d'un **contexte** précis. Chaque entreprise dispose d'un contexte qui lui est propre. Les éléments formant le contexte de l'entreprise, bien que n'étant pas de la responsabilité de cette dernière, entretiennent avec elle une relation privilégiée ou

[6] Dérivé de entreprendre, daté de environ 1430-40 dans le sens de « prendre entre ses mains ».
Aux environs de 1480 il prit le sens de « prendre un risque, relever un défi, oser un objectif ».

nécessaire et ont un impact significatif sur ses activités. Ce qui laisse facilement percevoir que l'étude d'une entreprise, pour être complète doit tenir compte de nombreux aspects spécifiques à l'entreprise mais aussi à son environnement.

Finalement, en ce qui concerne le niveau de granularité de ce que représente l'entreprise au sein de la démarche d'architecture, notons que le terme entreprise utilisé dans ce contexte peut tout aussi bien être utilisé pour désigner une entreprise complète, en englobant tous ses systèmes d'informations, ou un domaine métier spécifique de l'entreprise.

Architecture d'entreprise

L'architecture lorsqu'elle s'applique au domaine de l'entreprise peut être définie simplement en rappelant la définition de l'architecture proposée par le standard international ISO/IEC/IEEE. Cependant, en considérant l'ampleur qu'occupe la discipline d'AE dans le paysage des entreprises et de leurs SI, et compte tenu du nombre important d'acteurs impliqués dans ce domaine et du nombre de solutions proposées, il semble légitime et nécessaire d'étudier d'autres définitions traitant de l'AE afin de correctement en saisir le éléments primordiaux.

L'AE est :

« Une base stratégique d'informations sur les actifs, qui définit l'entreprise, les informations nécessaires pour faire fonctionner l'entreprise, les technologies nécessaires pour soutenir les opérations commerciales et les processus de transition nécessaires à la mise en œuvre de nouvelles technologies afin de répondre aux besoins changeants des entreprises. » [7]

[USGov]

« L'architecture d'entreprise est une discipline qui soutient de façon pro-active et holistique les entreprises à faire face aux forces perturbatrices, ceci en identifiant et en analysant l'exécution du changement vers la vision et les résultats souhaités. L'architecture d'entreprise crée de la valeur en documentant le métier et l'IT, ceci accompagné de recommandations « clé-sur-porte » pour permettre l'ajustement des politiques et des projets, afin d'atteindre les résultats opérationnels ciblés. » [8]

[GartnerGroup]

« Une description de l'organisation opérationnelle de l'entreprise associée à celle du système d'information supportant les dites opérations. Ceci en utilisant des concepts d'architecture (IT) au niveau le plus abstrait d'une organisation. Une Architecture d'entreprise s'applique sur les canaux de communications existants ainsi que sur la conduite des changements et comprend typiquement un organigramme fonctionnel, une cartographie du système d'information, l'identification des opportunités d'améliorations stratégiques ainsi que l'identification d'activités de transformation à une plus grande échelle. »

[AGF, 2014]

« L'architecture d'Entreprise est la logique d'organisation des processus métier clés et des capacités informatiques en reflétant les exigences d'intégration et de normalisation du modèle de fonctionnement de l'entreprise. » [9]

[MIT]

[7, 8, 9] Traductions personnelles

Bien que cette liste de définition ne soit pas exhaustive elle permet de découvrir, au travers du prisme de certains acteurs importants en la matière, les principaux aspects que recouvre une architecture d'entreprise ainsi que les objectifs visés par celle-ci.

On peut constater que l'architecture d'entreprise au travers de ces différentes définitions peut couvrir des périmètres plus ou moins grands, ceci avec des objectifs parfois différents. A première vue, cela pourrait laisser penser que le sujet n'a pas encore atteint un grand degré de maturité. Cependant, il faut tenir compte du point de vue de chacun des acteurs cités afin de pouvoir justifier en grande partie cette hétérogénéité. En effet, les acteurs cités ont construit leur définition de l'architecture d'entreprise en se basant sur leurs propres besoins et expériences en la matière, et donc en tenant compte des considérations qu'ils auront jugées nécessaires.

Si l'on devait tenter de fournir une définition de l'AE en essayant de retenir les éléments primordiaux et communément partagés par un ensemble des définitions, nous pourrions définir l'AE comme :

« Un ensemble cohérent, de principes, de méthodes et des modèles qui sont utilisés dans la conception et la réalisation de l'organisation, des processus, des systèmes d'information et de l'infrastructure de l'entreprise, ceci et tenant compte de l'environnement au sein duquel elle évolue.

L'architecture d'entreprise doit permettre de raisonner de manière globale à propos de l'entreprise et de son système d'information et permettre de planifier les phases de transitions nécessaires aux évolutions désirées par l'entreprise.

L'architecture d'entreprise vise l'ensemble des fonctions de l'entreprise (pilotage, métier et support). Elle entreprend de réunir l'ensemble des acteurs pour les amener à dialoguer, à coopérer et à s'approprier leur propre système d'information pour créer de la valeur et répondre à leurs exigences métier. Cette démarche s'accompagne de la mise en place d'une gouvernance, avec des responsabilités et rôles pour chacun des acteurs. »

Objectifs poursuivis par l'architecture d'entreprise

Le but de l'architecture d'entreprise est :

- **optimiser l'héritage** des processus métier, souvent fragmentés, au travers de l'entreprise.
- **fournir** un contexte stratégique encadrant les besoins de l'entreprise et leur évolution.
- **soutenir** l'exécution de la stratégie d'entreprise.
- permettre **une gestion et une exploitation optimale** du système d'information.
- **aligner** le système d'information afin qu'il corresponde à la stratégie d'entreprise.
- **maitriser** la complexité des processus et systèmes de l'entreprise.
- **combler** le déficit de communication entre les architectes et les autres parties prenantes.

Lorsque l'on regarde la façon d'utiliser la technologie pour ajouter de la valeur à l'entreprise, nous avons besoin de réponses aux questions suivantes :

- Quels sont les objectifs généraux de l'organisation ?
- Comment l'organisation réalise-t-elle ses processus métier ?
- De quelle manière les processus opérationnels de l'organisation sont-ils liés ?
- Quels processus semblent susceptibles d'être améliorés grâce à la technologie?

Coûts d'une architecture d'entreprise

Typiquement, la charge de création d'une architecture d'entreprise est dominée par trois facteurs :

- Les délais nécessaires afin d'effectuer une analyse complète.
- Le grand nombre de personnel interne absorbés par le processus d'architecture d'entreprise.
- L'armée de consultants coûteux mais nécessaires pour naviguer au travers des méthodologies complexes.

1.1.2 Historique de l'architecture d'entreprise

Ce bref historique de l'AE aborde les événements marquant de son histoire. La Figure 2 synthétise ces éléments sur une ligne du temps :

- **1960 : BSP** – *Business System Planning* (IBM)

  Premier travaux de formalisation des documents architecturaux par P. Duane Walker, qui continuera ses travaux en tant que directeur d'IBM. Ceci donna lieu à la Méthodologie BSP qui représente la première approche d'AE, se penchant sur des aspects tels que : la concrétisation d'opportunités grâce à la technologie, la production de plans de migration technologique, le support à la décision et le support du développement.

- **1986** : ***PRISM EA***

  Publication de PRISM EA par l'équipe de recherche de Dr. Michael Hammer, Dr. Thomas H. Davenport, et James Champy. PRISM EA est le premier cadre d'architecture d'entreprise publié dans ce que l'on considère comme la compréhension moderne de ce concept. Le cadre PRISM EA organise une description architecturale en seize catégories en fonction de quatre domaines (organisation, données, applications et infrastructure) et quatre types (inventaire, les principes, les modèles et les normes).

- **1987** : ***Framework for Information Systems Architecture*** (J.A. Zachman)

  Publication au Journal IBM Systems d'un article intitulé « Un cadre d'architecture pour les systèmes d'information ». Dans ce document J.A. Zachman fait évoluer BSP en proposant la première version de son cadre d'architecture et pose des bases solides en explicitant les défis et la vision des architectures d'entreprise. L'objectif poursuivi par cet article était de permettre la gestion de la complexité des systèmes de plus en plus distribués. La vision de Zachman étant que la valeur de l'entreprise et de l'agilité pourraient mieux être rencontrés par une approche holistique de l'architecture des systèmes qui exprimerait explicitement chaque question importante selon plusieurs perspectives (le Quoi, le Qui, le Où, le Comment et le Pourquoi). Zachman décrit de cette façon une approche multi-perspectives de l'architecture des systèmes.

- **1994** : **TAFIM** – *Technical Architecture Framework for Information Management* (DoD)

  Publication par le département américain de la Défense (DoD) de TAFIM (Technical Architecture Framework for Information Management), un cadre d'architecture destiné à la gestion de l'information. La pensée de Zachman semble avoir fortement influencée cette approche.

- **1995** : **TOGAF** – *The Open Group Architecture Framework* (Open Group)

Sélection par l'Open Group de TAFIM comme base pour le développement de TOGAF (The Open Group Architecture Framework). La démarche proposée par TOGAF est l'adoption d'une vue stratégique, construite à l'échelle de l'entreprise mais également orientée vers la technologie. L'un de ses principaux objectifs est de permettre la rationalisation des domaines informatiques, le plus souvent désordonnés. La première version de TOGAF verra le jour au début de l'année 1995.

- **1999** : **FEAF** - *Federal Enterprise Architecture Framework* (CIO)

  En 1998, le Conseil Américain Fédéral (CIO) initie son premier grand projet du genre, le cadre d'architecture d'entreprise fédérale FEAF (Federal Enterprise Architecture Framework). La version 1.1 de ce cadre a été publiée en Septembre 1999. Le cadre d'architecture FEAF décrit une approche, y compris les modèles et les définitions, pour développer, documenter et décrire l'architecture des segments fonctionnels multi-organisationnelles du gouvernement fédéral. Dans la continuité de l'approche du cadre de Zachman, le FEAF propose des modèles pour décrire le métier, les données, les applications et la technologie. Sa mise en œuvre au sein du gouvernement fédéral américain sera imposée par la loi nommée Clinger/Cohen qui conduira à de gros investissements, mais qui ne rencontrèrent pas le succès escompté.

- **2002** : **FEA** – *Federal Enterprise Architecture* (OMB)

  Au début des années 2000 se fit progressivement un transfert de compétence entre le CIO et le Bureau de la gestion et du budget (OMB) en ce qui concerne l'architecture d'entreprise fédérale américaine. En 2002, l'OMB réévalue et renomme la méthodologie FEAF sous la dénomination FEA (Federal Enterprise Architecture). FEA est la dernière tentative du gouvernement fédéral d'unir ses innombrables agences et fonctions sous une seule et même architecture d'entreprise commune.

- **2005** : *Pratique d'architecture d'entreprise* (Garner Meta)

  Suite à l'achat en 2005 de Meta Group par Garner, Garner Meta passa une année à examiner l'expérience d'architecture d'entreprise et les méthodologies en analysant les propositions issues des deux compagnies. Ceci donna lieu à la pratique d'architecture d'entreprise de Gartner. Cette dernière méthodologie est un peu différente des précédentes, ce n'est pas une taxonomie (comme Zachman), un processus (comme TOGAF), ou une méthodologie complète (comme FEA). Au lieu de cela, la proposition de Gartner se présente comme une pratique qui s'appuie essentiellement sur une compilation d'expériences et des spécialistes qualifiés, qui ont développé une communauté encourageant la collaboration et les bonnes pratiques dans le domaine de l'AE. La bibliothèque documentaire de Garner propose des notes de recherche décrivant leur pratique d'architecture d'entreprise. Par exemple, il existe "Gartner Enterprise Architecture Process: Evolution 2005" et "Gartner Enterprise Architecture Framework: Evolution 2005". Cependant, ces documents contiennent peu d'informations descriptives et datent de la fin de l'année 2005. Gartner soutient que ses bonnes pratiques sont intemporelles et qu'elle continue à les augmenter au besoin, sans pour autant proposer de formalisations plus exhaustives de ses pratiques.

- **2006** : **EAAS** – *Enterprise Architecture As Strategy* (MIT)

  Publication en 2006 par le Centre de recherche des systèmes d'information du MIT d'un livre intitulé : "Enterprise Architecture As Strategy". Cet ouvrage accorde une grande importance à la nécessité pour les architectes d'entreprise de se concentrer sur les processus métier de l'entreprise. La justification est que les entreprises excellent parce qu'elles ont décidé quels processus métier doivent être soutenus par l'informatique. L'importance de l'engagement des gestionnaires d'entreprises dans la démarche d'architecture d'entreprise est également mise en avant.

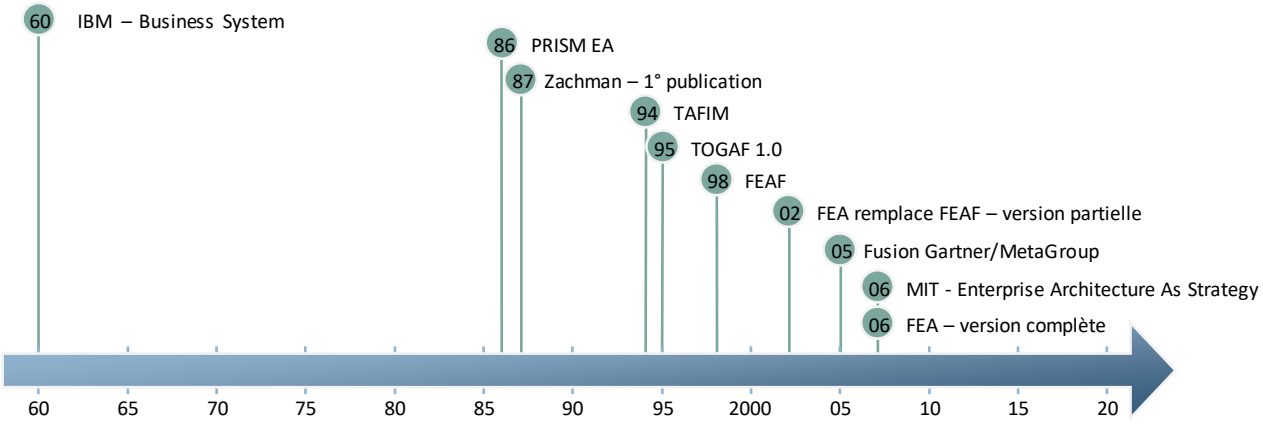

Figure 2 : Ligne du temps de l'AE

1.1.3 Les domaines de l'architecture d'entreprise

Afin de définir les domaines de l'architecture d'entreprise, nous allons nous baser sur les travaux menés par Stephen Spewaks en 1993 (EAP – Enterprise Architecture Planning) qui subdivisent l'architecture d'entreprise selon quatre domaines d'architecture majeurs, ceci tel que présenté dans la Figure 3.

Les domaines d'architecture d'entreprise sont des représentations partielles d'un système entier et leur objectif est de répondre aux préoccupations de plusieurs parties prenantes.

Chacun de ces domaines se concentre sur une perspective importante de l'entreprise et de son système d'information.

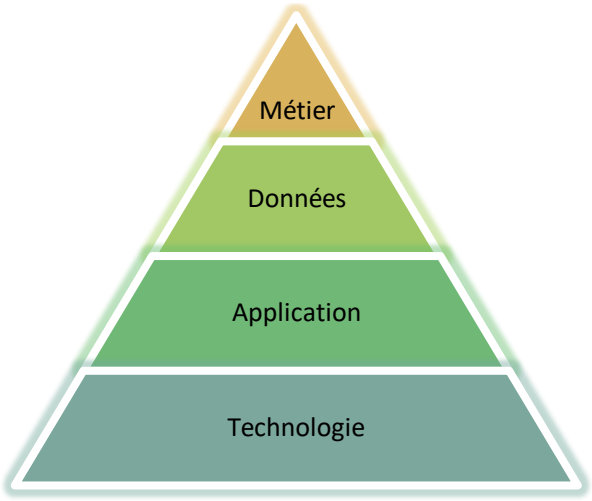

Figure 3 : Couches de l'architecture d'entreprise [NILES, 2006]

Les quatre sections suivantes traitent des différents domaines de l'AE et sont le résultat d'une synthèse réalisée sur base d'éléments issus des sources suivantes : [TOGAF9.1], [EADomainsWiki] et [RealIRM]

Domaine d'architecture métier

Le terme « architecture métier » est utilisé pour désigner une description architecturale composée de modèles qui permettent une compréhension commune de l'entreprise. La caractéristique principale de l'architecture métier est qu'elle modélise les aspects du monde réel d'une entreprise ainsi que leurs interactions. Elle se concentre sur la définition et l'analyse des préoccupations de ce que fait l'entreprise, de son fonctionnement, de son organisation et de ses valeurs.

L'architecture métier peut être utilisée pour concevoir des structures et des processus compétitifs, tirer parti des atouts existants et identifier les opportunités d'investissement potentielles qui feront progresser l'atteinte des objectifs de l'entreprise et qui stimuleront l'innovation. Cette démarche permet de guider l'entreprise sur base de plans, afin que cette dernière puisse prendre des décisions commerciales motivées et à en orienter les implémentations.

L'effort d'architecture métier peut être mené seul ou dans le cadre d'une architecture d'entreprise. Bien qu'une pratique d'architecture d'entreprise dans le passé ait porté principalement sur les aspects technologiques du changement, la pratique évolue rapidement pour utiliser une approche rigoureuse de l'architecture métier pour aborder les aspects organisationnels et de motivation du changement au sein de l'entreprise. L'alignement entre l'IT et l'architecture métier est un alignement naturel de deux disciplines connexes. L'architecture métier représente une entreprise en l'absence de toute architecture informatique. L'architecture d'entreprise constitue un cadre global pour l'architecture métier de l'entreprise et de son système d'information.

Domaine d'architecture des données

L'architecture des données est utilisée pour définir les modèles de données de l'entreprise et les interactions relatives aux données.

L'architecture des données se concentre donc principalement sur la structure des données, ceci comprenant autant les données stockées que les données en mouvement.

Comme pour tous les autres domaines d'architecture, l'architecture des données est une perspective essentielle pour réaliser une démarche d'architecture d'entreprise. Ce domaine d'architecture doit décrire tous le savoir et l'information nécessaire à l'entreprise pour mener à bien ses processus métiers. Les données sont documentées selon différentes perspectives et les éléments nécessaires à leur compréhension et leur gouvernance sont également requis. Voici une liste non exhaustive présentant l'information généralement documentée au sein de l'architecture des données :

- Principes et politiques de gestion des données
- Patterns de structuration de l'information
- Modélisation des objets métier
- Modélisation des données applicatives
- Schéma de stockage persistant des données applicatives

En résumé, l'architecture des données permet donc de concevoir des structures de données et de contrôler les flux de données au sein du système d'information.

Au cours de la définition de l'état cible d'une entreprise, l'architecture des données brise un sujet au niveau atomique pour ensuite le restituer à la forme souhaitée. A cette fin l'architecte de données passe par trois processus architecturaux traditionnels qui sont :

- Conceptuel - représente toutes les entités métier et leurs relations.
- Logique - représente la logique des relations entre les entités, et leurs attributs.
- Physique - représente les données pour un type spécifique de technologie.

Domaine d'architecture des applications

L'architecture des applications est utilisée pour définir les applications logicielles supportant les processus métier ainsi que les capacités métier procurées par les services applicatifs.

Pour ce faire l'architecture applicative se concentre sur la description des comportements des applications et de quelles façons ces dernières interagissent entre elles ainsi qu'avec les utilisateurs.

L'architecture applicative ne se concentre pas sur le fonctionnement interne des applications mais plutôt sur la manière dont ces dernières manipulent et produisent l'information. La motivation pour les applications est définie en documentant la couverture fonctionnelle de ces dernières, ceci à l'aide des éléments identifiés dans le cadre du domaine d'architecture métier.

Parmi les cadres d'architecture généraux, la combinaison de l'architecture des données et de l'architecture applicative semble naturelle, ce qui conduit majoritairement les cadres d'architecture d'entreprise à rassembler ces deux domaines d'architecture sous un domaine unique. Cette combinaison représente alors le système d'information complet de l'entreprise. Ce domaine se situe sous le domaine métier et au-dessus du domaine technologique.

Domaine d'architecture de la technologie

L'architecture de la technologie est utilisée pour définir le hardware et le software et tout autre élément d'infrastructure qui est jugé nécessaire dans le but de soutenir le développement et l'hébergement des applications détaillées dans le domaine d'architecture applicative.

1.1.4 Classification des cadres d'Architecture d'Entreprise

Afin de donner forme à l'architecture d'entreprise telle que discutée jusqu'à présent, voici un aperçu d'un certain nombre de cadre d'architecture d'entreprise bien connus et utilisés à ce jour. Ces cadres d'architecture d'entreprise permettent de structurer et de décrire une architecture d'entreprise en identifiant et en abordant différents points de vue architecturaux (=domaines) ainsi que les techniques de modélisation associés. Cependant ces cadres d'architecture d'entreprise ne fournissent pas toujours les concepts de modélisation réelle, même si certains cadres sont étroitement liés à un langage de modélisation ou un ensemble de langages.

La plupart des cadres d'architecture sont assez précis pour permettre d'identifier les éléments qui devraient faire partie d'une démarche d'architecture d'entreprise particulière. Et pour garantir la qualité de l'architecture d'entreprise au cours de son adoption et de son cycle de vie, l'utilisation de processus ou de recommandations dédiés à la mise en place d'une architecture d'entreprise sont nécessaires. Lorsque de tels processus sont disponibles, ils aident les architectes à parcourir toutes les phases du cycle de vie des architectures d'entreprise.

Un certain nombre de cadres d'architecture sont apparus ou ont évolués ces dernières années. Une classification de ces cadres pourrait être réalisée selon quatre grands types :

- **Les cadres de contenu** (=taxonomie), organisent et décrivent les concepts (processus, acteur, application, …) concernés ou faisant partie du SI, ainsi que leurs relations.

  Exemple : Le **cadre d'AE de Zachman**, défini plus précisément comme une taxonomie s'apparente au 1er type de cadre.

- **Les cadres de processus**, organisent et décrivent formellement les activités nécessaires à la réalisation d'une démarche d'architecture d'entreprise.

  Exemple : **The Open Group Architecture Framework** (TOGAF version < 9) était initialement uniquement une méthode de développement d'architecture (sans cadre de contenu), ce qui correspond au 2ème type de cadre.

- **Les cadres de contenu et de processus,** organisent et décrivent le contenu et le processus, tel que présenté dans les deux points précédents. Ce type de cadre d'AE à l'avantage de proposer un bon degré d'homogénéité entre la taxonomie et le cadre de processus.

  Exemple : **Federal Enterprise Architecture** (FEA) pouvait, jusqu'à la parution et l'adoption de TOGAF 9, être considérée comme la méthodologie la plus aboutie puisqu'elle associe un cadre de processus et un cadre de contenu, ce que décrit le 3ème type de cadre.

- **Les cadres plus ouvert, s'appuyant sur un savoir-faire**, ce type de cadre différant ne propose pas de cadre de contenu ou de processus particulier, mais se présente plutôt comme une pratique capitalisant une certaine expertise dans le domaine de l'AE.

Exemple : **Gartner** est le cadre atypique de 4ème type : une pratique nécessitant la mise en œuvre de compétences opérationnelles basées sur l'expérience, la formation et la communication. Ce type de cadre ne recommande pas de cycle ou de contenu prédéfini mais prône d'avantage une approche personnalisée selon les sujets d'étude.

Concernant d'éventuelles données relatives aux taux d'utilisation des différents cadres d'AE, malheureusement les recherches à ce sujet réalisées durant la rédaction de ce mémoire se sont montrées infructueuses. Malgré les nombreuses recherches dans diverses bases de connaissances il n'a pas été possible de produire une image du taux d'utilisation des cadres d'AE les plus populaires.

Néanmoins, depuis 2013 il est communément admis que, sur base du nombre de certifications octroyées, TOGAF est devenu le cadre d'architecture d'entreprise le plus populaire. Ceci à tel point que certains supposent même que TOGAF définit désormais la discipline d'AE.

1.2 TOGAF – The Open Group Architecture Framework

Ce chapitre va nous permettre de découvrir TOGAF et de quelle manière ce cadre d'architecture d'entreprise s'organise pour proposer une solution générique soutenant la démarche de définition et de gestion d'une architecture d'entreprise.

Cette présentation de TOGAF doit nous permettre d'en découvrir l'essentiel afin d'identifier dans la partie suivante les éléments pertinents et nécessaires à l'intégration du style d'architecture DDD au sein de TOGAF.

Remarque :

Toutes les références faites à des éléments TOGAF dans ce mémoire se basent sur la version 9.1 de ce dernier. L'ensemble du contenu du document TOGAF 9.1 est disponible à l'adresse suivante : http://pubs.opengroup.org/architecture/togaf9-doc/arch/index.html.

1.2.1 Présentation

TOGAF, acronyme de « The Open Group Architecture Framework », est le cadre d'architecture d'entreprise élaboré par le consortium international nommé « Open Group ».

Aujourd'hui reconnu comme un standard industriel, TOGAF propose un ensemble méthodologique complet, à usage libre, basé sur des modèles génériques et des bonnes pratiques, afin de piloter la transformation de l'entreprise et de professionnaliser la fonction d'architecture d'entreprise.

TOGAF a été continuellement amélioré depuis le milieu des années 1990 par différentes personnes appartenant à un certain nombre de départements informatiques d'importantes sociétés, ainsi que par des fournisseurs de conseils ou de solutions informatiques.

Il s'appuie sur trois concepts fondamentaux que sont :

- le cadre de processus
- le cadre de contenu
- le cadre de capacité

1.2.2 Historique

Cet historique de TOGAF nous permet de découvrir son évolution au fil du temps. Les changements amenés par ses différentes versions montrent l'évolution de la discipline d'architecture d'entreprise promue par l'Open Group. Les éléments présentés ci-après sont issus des notes de publication de TOGAF9[10] et d'une publication de l'association ADELI[11], et sont synthétisés à la Figure 4.

- **1995 :** TOGAF Version 1 (proof of concept)

  Cette première version a été construite sur les bases posées par TAFIM (Technical Architecture Framework for Information Management), elle constitue le premier cadre d'architecture d'entreprise publié par Open Group. Cette première version concernait principalement le domaine d'architecture technique. TOGAF Version 1 démontre la faisabilité du projet.

- **1996** : TOGAF Version 2 (proof of application)

  Cette seconde version va plus loin que la version précédente en consolidant la méthode par une démonstration de son applicabilité au moyen de diverses mises en œuvres concrètes.

- **1997** : TOGAF Version 3 (building blocks)

  Cette version complète la méthode avec des solutions pratiques sous forme de blocs d'architecture réutilisables.

- **1998** : TOGAF Version 4 (enterprise continuum)

  Cette version introduit le « continuum d'entreprise »,, un modèle permettant de répondre à la question de l'organisation de l'information sur l'architecture au sein de l'entreprise en structurant un référentiel virtuel.

- **1999** : TOGAF Version 5 (ADM)

  Cette version introduit la méthode de développement de l'architecture : le cycle ADM (Architecture Development Method).

- **2000 :** TOGAF Version 6

  Cette version poursuit l'évolution de TOGAF en se référant aux définitions et principes de la norme ANSI/IEEE 1471-2000 ainsi qu'aux travaux du DoD (Département de la Défense américaine).

- **2001** : TOGAF Version 7 (« Technical Edition »)

  Cette version, toujours axée sur l'étude du domaine d'architecture technique, permet à TOGAF d'atteindre un niveau de maturité permettant la mise en œuvre de certifications de personnes.

[10] http://pubs.opengroup.org/architecture/togaf9-doc/arch/

[11] http://www.adeli.org/document/593-l81p09pdf

- **2002** : TOGAF Version 8 (« Enterprise Edition »)

  Cette version de TOGAF utilise la même méthode de développement d'architecture d'entreprise que celle employée dans les versions précédentes mais en étend le champ d'application aux autres domaines de l'entreprise et de cette façon laisse clairement apparaître les quatre domaines de l'architecture d'entreprise (métier, applicative, donnée, technologique) au sein du cadre d'architecture. A côté de cela, cette version clarifie la démarche en y distinguant deux axes complémentaires que sont la description de l'architecture et sa gestion.

  - **2006 :** TOGAF Version 8.1.1 (Mise à jour qualitative)

Le rythme de production s'est alors ralenti permettant la mise en œuvre de certifications sur des bases stables.

- **2009 :** TOGAF Version 9

.

  Cette version de TOGAF est la dernière version majeure publiée à ce jour. Elle complète la démarche en apportant une série d'outils pratiques et d'améliorations diverses dont :

  - Amélioration de la modularité de la structure afin d'en faciliter l'utilisation.
  - Ajout du cadre de contenu améliorant la cohérence et l'homogénéité de la démarche.
  - Améliorations des recommandations sur l'adoption de TOGAF au sein des entreprises
  - Mise à disposition de nouveaux outils et d'une grande quantité d'exemples.
  - Considération explicite des styles architecturaux : (SOA, Enterprise Security Architecture).

  - **Décembre 2011** : TOGAF 9.1 (Mise à jour qualitative)

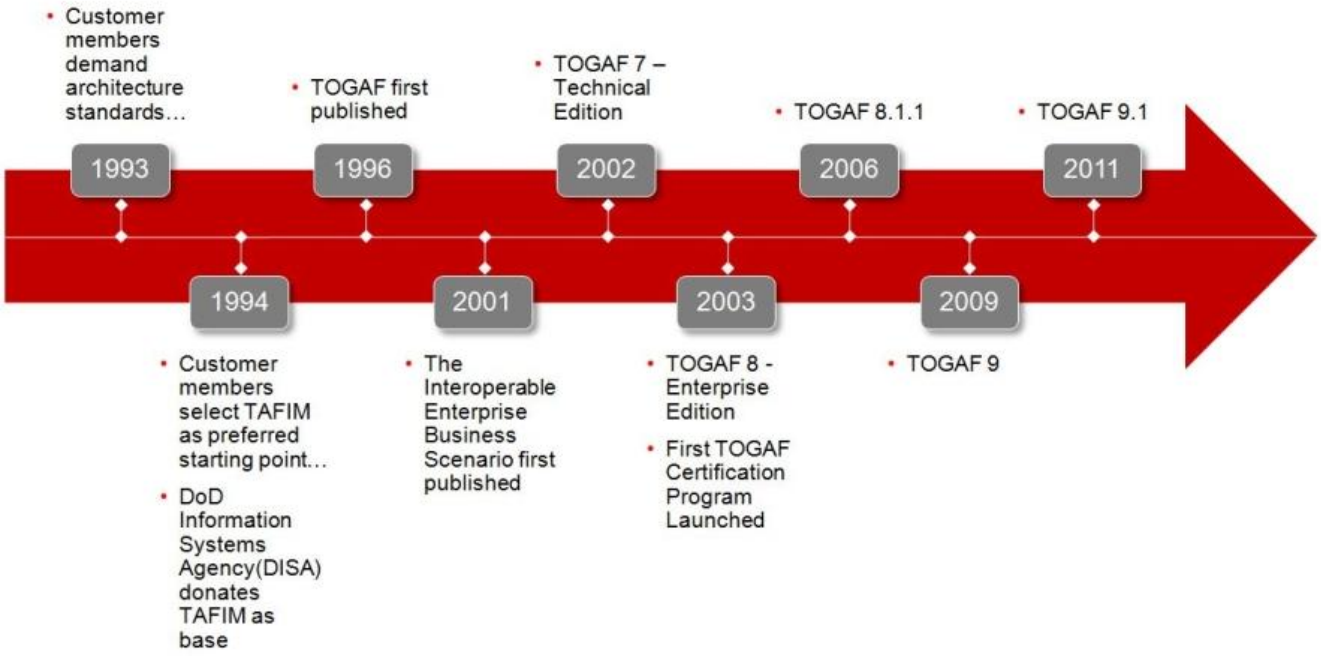

Figure 4 : TOGAF – Ligne du temps [ArchiMetric]

1.2.3 Le contenu de TOGAF

Dans la pratique TOGAF se présente sous la forme d'un unique document de référence et d'un site web dédié (voir **[TOGAF9.1]**). La Figure 5 représente de manière synthétique la structure générale de ce document de référence :

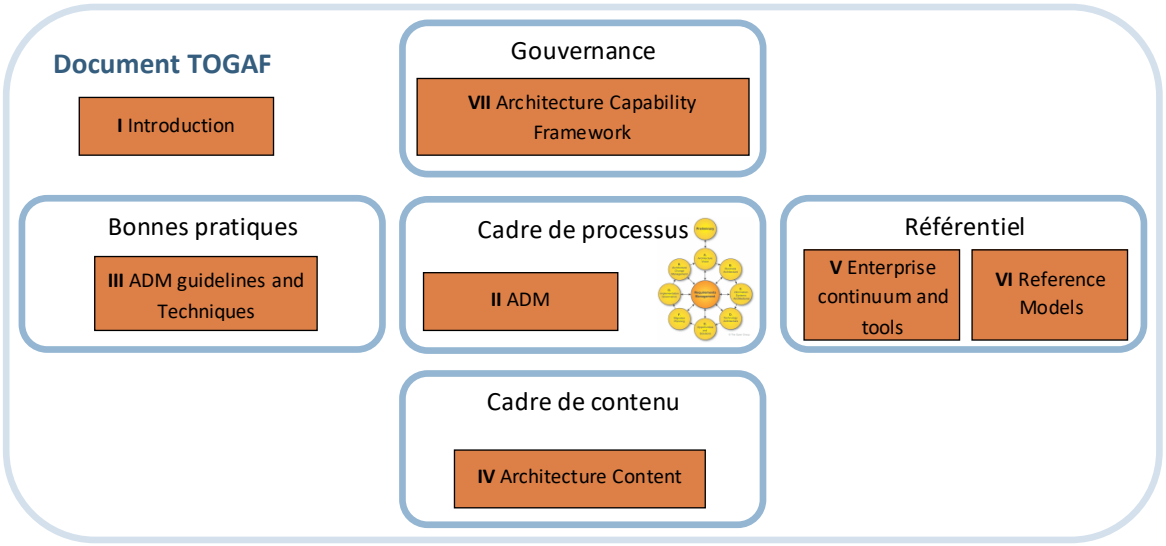

Figure 5 : TOGAF – Structure générale du document TOGAF [TOGAFGuideDePoche, 2010]

La **partie I** : **Introduction** est une présentation générale du cadre d'architecture TOGAF.

La **partie II : ADM** s'impose comme le point d'entrée principal du document. Il décrit le cycle et ses différentes phases.

La **partie III : ADM guidelines and Techniques** est un ensemble de recommandations et de techniques permettant de soutenir l'application de l'ADM.

La **partie IV : Architecture Content** est consacrée à la définition des éléments tangibles utilisés dans le cadre de la démarche d'architecture d'entreprise.

La **partie V : Enterprise Continuum and Tools** fourni des recommandations relatives à l'organisation et la classification des éléments d'architecture, de structuration du référentiel d'architecture[12] et documente les outils de création et de gestion des artefacts d'architecture sélectionnés.

La **partie VI : Reference Models** propose deux modèles de référence, à savoir le Modèle de Référence Technique (TRM) et le Modèle de Référence d'Infrastructure d'Information Intégrée (III-RM).

La **partie VII : Architecture Capability Framework** traite de la gouvernance de l'architecture, ce qui comprend la gestion du référentiel.

[12] Le référentiel d'architecture contient le contenu produit par la méthode ADM et est structuré selon le continuum d'entreprise

TOGAF propose une structuration de son document de référence qui décrit de façon indépendante les spécifications de ses différentes parties. L'objectif poursuivi par cette approche est de permettre une considération de différents domaines de spécialisation de façon isolée. Bien que toutes les parties de TOGAF soient complémentaires, cette approche permet une adoption partielle et évolutive des différentes parties du document de référence TOGAF au sein d'une organisation.

La Figure 6 présente la structure des parties de TOGAF et leurs relations au sein d'une organisation.

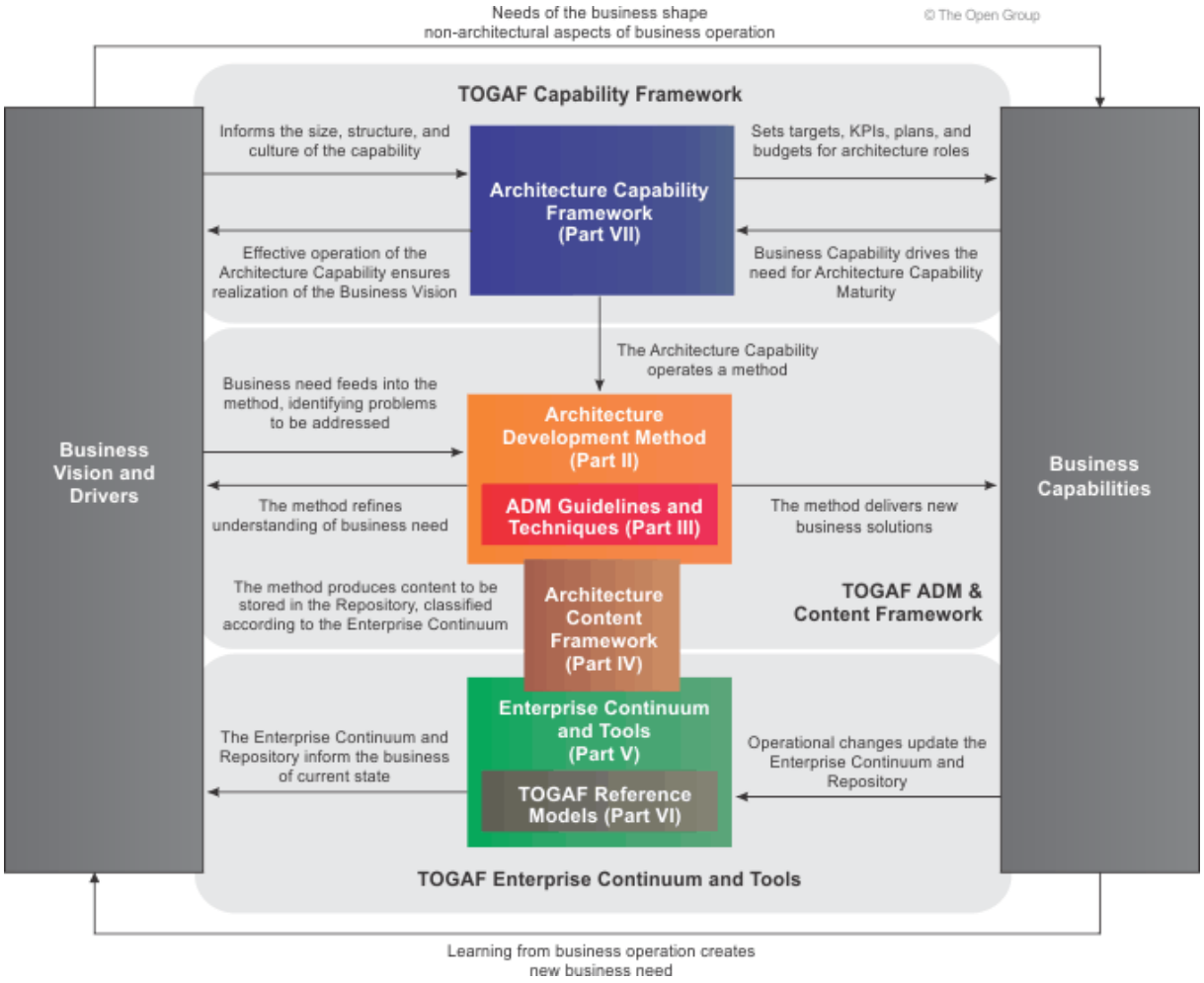

Figure 6 : TOGAF – Synoptique du contenu de TOGAF [TOGAF9.1]

PARTIE I : Introduction (Introduction)

Cette partie propose une introduction de haut niveau aux concepts clés de l'architecture d'entreprise et à TOGAF. La lecture de cette partie est pertinente pour toute personne non initiée à l'architecture d'entreprise et désireuse de découvrir cette pratique et les solutions proposées par TOGAF.

Cette partie du document TOGAF se compose des sections suivantes :

- **Introduction**, fournit une introduction très générale au domaine de l'AE et à TOGAF.
- **Core Concepts**, présente brièvement les concepts de base essentiels de TOGAF.
- **Definitions**, est une liste de termes et de définitions applicables dans le cadre de TOGAF.
- **Release Notes**, présente l'ensemble des nouveautés de TOGAF et fournit des outils permettant d'établir des correspondances avec les éléments de la version précédente de TOGAF (dans notre cas TOGAF 8.1.1).

PARTIE II : ADM (Architecture Development Method)

Le **cadre de processus** permet de développer une architecture d'entreprise répondant aux besoins métiers.

> « La Méthode de Développement d'Architecture est la partie centrale de TOGAF. Une approche par étape afin de développer et utiliser une architecture d'entreprise. »
>
> **[OGF, 2012]**

Le cycle ADM

Le cycle ADM est le cadre de processus proposé par TOGAF. Ce cadre est un des piliers sur lequel repose la démarche d'architecture d'entreprise proposée par l'Open Group.

La méthode ADM se compose de 8 phases essentielles (de A à H) et de deux autres phases particulières que sont la phase préliminaire ainsi que la phase de gestion des exigences.

La Figure 7, l'une des plus citées lorsque l'on parle de TOGAF, synthétise cette démarche.

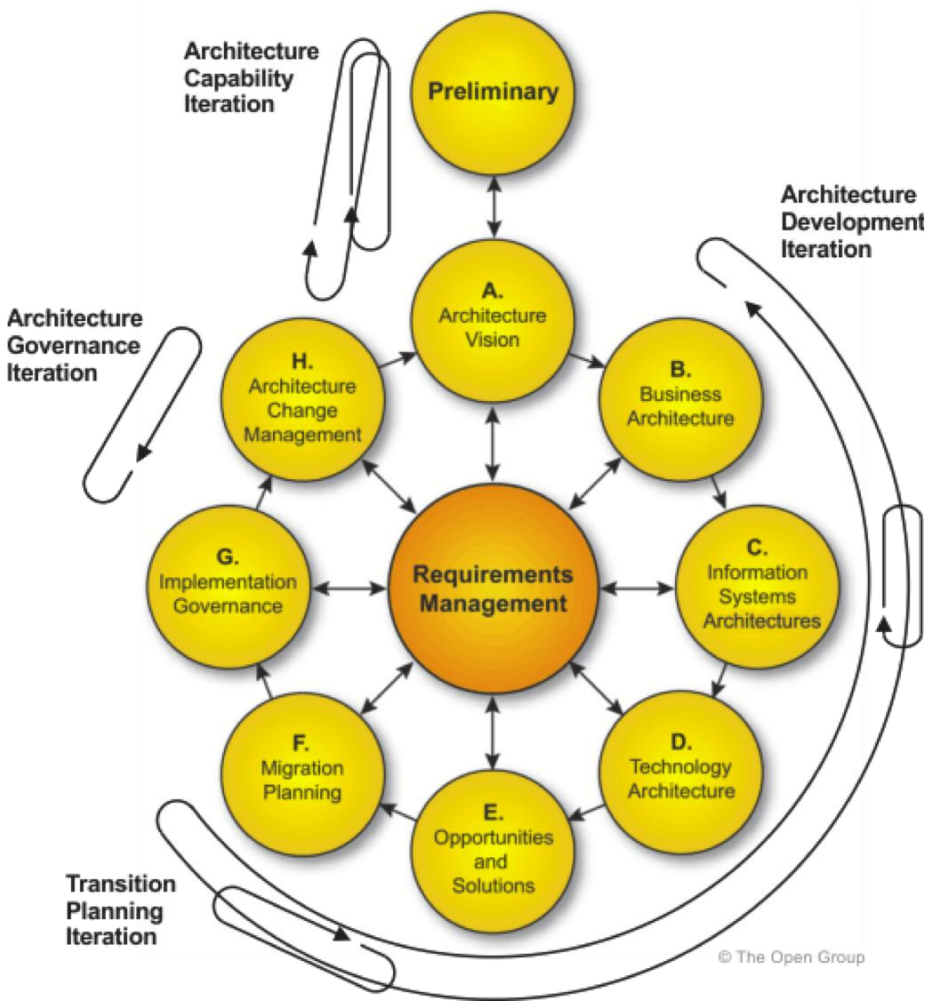

Figure 7 : TOGAF – Cycle ADM [TOGAF9.1]

Bien que l'enchaînement de phases soit décrit de manière strictement séquentielle (de A à H), celui-ci peut être adapté en fonction du contexte d'application. Le plus souvent cette adaptation se fait sous forme d'itérations au sein du cycle ADM.

De façon plus générale, ce schéma doit être considéré comme une structure de référence et non comme une structure intangible. Dans la pratique, l'ajustement ou la remise en cause des résultats issus des phases précédentes est une situation possible, voir souhaitable. Par exemple, la découverte de nouvelles contraintes, la modification ou le raffinement de certaines exigences peut faire apparaître des éléments nouveaux qui n'ont pas été exploités durant les phases précédentes, ce qui implique de devoir repasser par ces phases antérieures afin d'en modifier ou compléter le contenu.

Open Group propose un ensemble de guides de recommandation concernant la réalisation et l'adaptation du cycle ADM. Ces techniques sont décrites dans la partie III du document TOGAF : « ADM guidelines and Techniques ».

Les phases du cycle ADM

Phase préliminaire (Preliminary)

Cette phase ne fait pas réellement partie des itérations réalisées au sein du cycle ADM, ses activités sont essentiellement de nature transverse et sont d'avantage liées à la gouvernance générale de l'architecture d'entreprise et à l'adaptation du cadre d'architecture au contexte de l'entreprise. Cette phase a pour objectif de préparer l'entreprise à la réalisation des travaux d'architecture. Il s'agit de mettre l'entreprise en capacité de maîtriser la gestion et les transformations de son architecture.

Selon TOGAF, les actions a mener afin de déterminer la Capacité d'Architecture souhaitée par l'entreprise sont les suivantes :

- Examiner le contexte organisationnel au sein duquel se déroule la démarche
- Définir le niveau de maturité ciblé
- Définir et mettre en place le modèle organisationnel de la démarche d'AE
- Définir et mettre en place les processus et les ressources nécessaire à la gouvernance
- Définir les principes d'architecture
- Sélectionner et mettre en œuvre les outils supportant la Capacité d'Architecture

Finalement, TOGAF donne une bonne synthèse de cette phase sous la forme du : « où, qui, pourquoi, qui et comment » [TOGAF9.1].

Phase A : Vision de l'Architecture (Architecture Vision)

Cette phase est la première faisant partie de l'itération au sein du cycle ADM. Cette phase permet d'enrichir et de préciser les éléments issus de la phase préliminaire, de plus elle prépare les phases suivantes sous la forme d'une représentation générale de l'architecture initiale et cible.

Selon TOGAF, les objectifs poursuivis par cette phase sont :

- Documenter une vision de haut niveau concernant les capacités et la valeur commerciale fournie par la démarche d'AE proposée.

- Obtenir l'approbation nécessaire à la réalisation des travaux de mise en œuvre de l'architecture d'entreprise telle que définie.

La communication joue un rôle important dans cette phase, il s'agit en effet d'assurer qu'il existe un consensus entre toutes les parties prenantes à propos des orientations et des résultats attendus de la démarche d'AE.

Phases B, C, D : Architectures Métier/SI/Technologique (Business/IS/Technology Architectures)

Les trois phases suivantes se concentrent sur des domaines différents de l'entreprise mais partagent une même approche. Celle-ci consiste à étudier les architectures initiales et cibles pour ensuite mesurer l'écart entre les deux. Sur base de cet écart, une évaluation des impacts des évolutions requises peut être réalisée.

Phase B : Architecture Métier (Business Architecture)

Cette phase du cycle ADM se concentre sur l'étude du domaine d'architecture métier. Elle est une phase déterminante dans la mesure où c'est bien le métier qui pilote l'architecture sous toutes ses formes. La formalisation des éléments métiers est une précondition nécessaire à toutes constructions logiques ou physiques valides.

Phase C : Architecture des systems d'information (Information Systems Architecture)

Cette phase du cycle ADM se concentre sur l'étude des domaines d'architecture applicative et d'architecture des données. Il forme en quelques sortes une phase charnières entre l'expression des besoins métiers et leur traduction physique. Durant cette phase les constituants logiciels (application et données) soutenant l'automation et l'exécution des capacités métier son étudiées. Cette étude est réalisée en dehors de toute considération technologique.

Phase D : Architecture technologique (Technology architecture)

Cette phase du cycle ADM se concentre sur l'étude du domaine d'architecture technologique. Son objectif est d'établir une correspondance technique et physique avec les éléments issus de la phase précédente, et de définir de cette façon un ensemble cohérent de composants logiciels, d'infrastructures, de plateformes techniques, … permettant la mise en œuvre du système d'information de l'entreprise.

Phase E : Opportunités et Solutions (Opportunities and Solutions)

Cette phase du cycle ADM est une phase de consolidation des résultats obtenus durant les phases d'élaborations précédentes (B, C et D). Cet état consolidé constitue la base principale permettant l'élaboration des architectures de transitions nécessaires à l'atteinte de l'état

cible. Pour ce faire, cette phase tient compte également des capacités de l'entreprise et accorde une grande importance à ce que les architectures de transitions délivrent continuellement une plus-value métier. Notons encore que cette phase permet d'initier la création du plan d'implémentation et de migration qui sera par la suite complété durant la phase F.

Phase F : Planification de la Migration (Migration Planning)

Cette phase du cycle ADM se concentre sur la planification de la migration en contextualisant dans le cadre de l'entreprise (organisation du changement, équipe, coût, …) les actions nécessaires à la transition de l'état initiale à l'état cible. Pour ce faire elle finalise les feuilles de routes issues de la phase précédente et veille à ce que les plus-values métiers ainsi que les coûts liés aux changements soient clairement compris par les parties prenantes.

Phase G : Gouvernance de la mise en œuvre (Implementation Gouvernance)

Cette phase du cycle ADM permet de produire la version définitive des contrats d'architecture passés avec les projets d'implémentation et de réaliser ensuite, sur base de ces contrats des revues de conformité auprès des projets de mise en œuvre. Ceci afin d'assurer la concordance entres les développements et l'architecture d'entreprise définie.

Phase H : Gestion de la maintenance et des évolutions (Architecture Change Management)

Cette phase du cycle ADM est une phase de gestion de l'architecture d'entreprise mise en place. Elle évalue les demandes de modification impactant l'architecture d'entreprise afin de les prendre en charge.

La gestion des exigences (Requirements management)

Cette activité se présente comme une activité centrale et continue autour de laquelle s'articule le cycle ADM. On peut donc considérer que le cycle ADM est continuellement piloté par cette activité de gestion des exigences.

TOGAF considère que les exigences ne sont pas définies et figées à l'entrée d'un cycle mais prône plutôt une vision dynamique des exigences qui peuvent évoluer au cours de la réalisation des travaux d'architecture. La possibilité de prendre en compte le changement des exigences est une caractéristique essentielle du cycle ADM permettant à l'entreprise de s'adapter à l'incertitude et au changement.

TOGAF définit une exigence comme « un énoncé quantitatif d'un besoin métier pris en charge par un élément particulier ou un lot de travail donné ». Concrètement, l'ensemble des exigences expriment ce qui doit être mise en œuvre dans la démarche d'architecture d'entreprise, et inversement ce que l'on ne retient pas.

PARTIE III: Recommandations et techniques de l'ADM (ADM Guidelines & Techniques)

Cette partie de du document TOGAF propose un ensemble recommandations et de techniques permettant de soutenir l'application de l'ADM.

Les recommandations et techniques apportent un ensemble de lignes directrices aidant à l'adaptation et à l'exécution de l'ADM afin de permettre à ce dernier de faire face à différents scénarios spécifiques. Les recommandations et techniques suggérées concernent tant le cycle (par exemple l'intégration d'itérations) que des contraintes spécifiques (par exemple l'intégration de la sécurité), ou bien encore portent sur des tâches spécifiques de l'ADM (analyse d'écart, planning de migration, …)

Voici la liste des sujets actuellement traités au travers de la partie III : ADM Guidelines and Techniques :

- Itération dans l'ADM
- Paysage d'architecture
- Security Architecture
- SOA
- Principes d'architecture
- Gestion des parties prenantes
- Pattern d'architecture
- Scénarios commerciaux et objectifs commerciaux
- Analyse des écarts
- Techniques de planification de la migration
- Exigences d'interopérabilité
- Évaluation de la préparation à la transformation des entreprises
- Gestion de risque
- Planification basée sur la capacité

PARTIE IV : Cadre de contenu (Architecture Content Framework)

Le **Cadre de Contenu** (ou cadre conceptuel d'architecture) est consacré aux éléments de l'architecture et aux supports utilisés pour leur description. Son objectif est double : d'une part, il définit à l'aide d'un métamodèle de contenu les types de blocs de construction, leurs relations et leurs attributs, et d'autre part propose une structuration cohérente et standardisée permettant de classifier les éléments produits à l'aide du cadre de contenu, ceci en définissant leurs relations. Les artefacts produits à l'aide du cadre de contenu deviennent alors le résultat d'architecture d'entreprise.

Une définition précise des constituants du vocabulaire et des représentations employées et un élément indispensable afin d'obtenir une communication efficace entre les différentes parties prenants de la démarche d'architecture d'entreprise.

> *« Le contenu du cadre conceptuel d'architecture de TOGAF fournit une structure pour le contenu architectural qui permet de manière constante de définir, structurer et présenter les divers composants. »*
>
> **[OGF, 2012]**

L'utilisation du cadre de contenu permet, entre autres, de :

- Fournir une liste complète des résultats attendus par l'AE.
- Fournir un standard ouvert sur la manière dont les architectures doivent être décrites.
- Favoriser une meilleure intégration des résultats de l'AE dans le cas d'une adoption du cadre de contenu par toute l'entreprise.

Les interactions entre le métamodèle, les éléments de base, les diagrammes, les points de vue et les parties prenantes sont décrites dans la Figure 8 :

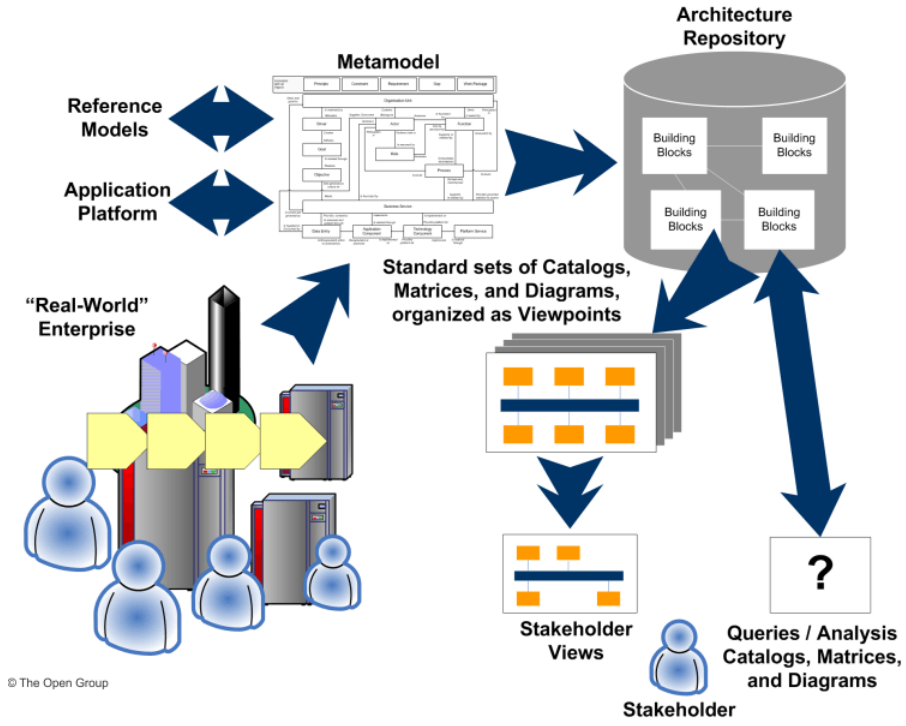

Figure 8 : TOGAF - Interactions entre Métamodèle, Building Blocks, Diagrammes et Parties prenantes [TOGAF9.1]

Le Métamodèle

Le métamodèle (=cadre de concept) définit les **éléments de base** utilisés pour la création de l'architecture. Ce métamodèle peut aussi être utilisé comme base à la configuration d'un outil de modélisation d'architecture d'entreprise.

« Un modèle qui décrit comment et avec quoi l'architecture sera décrite de façon structurée. »
[OGF, 2012]

Au niveau le plus élevé, le cadre de contenu est subdivisé en fonction des phases ADM de TOGAF :

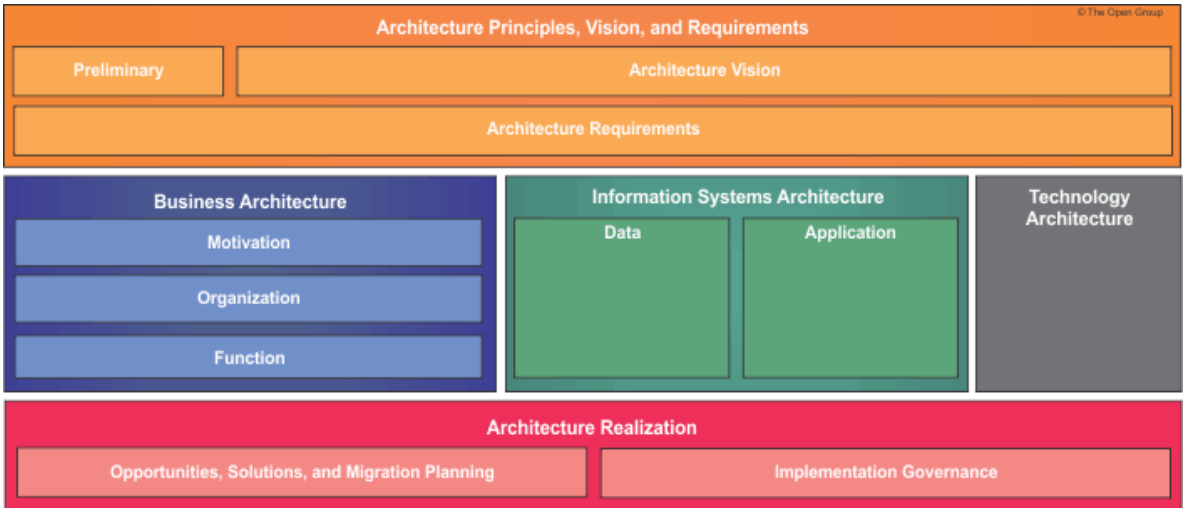

Figure 9 : TOGAF – Cadre de contenu [TOGAF9.1]

Subdivisions du métamodèle de base TOGAF :

Architecture Principles, Vision, and Requirements

Ces éléments de base sont destinés à capturer le contexte environnant des modèles d'architecture, ceci y compris les principes généraux de l'architecture, le contexte stratégique de réalisation de l'architecture et les exigences générées par l'architecture. Le contexte d'architecture est généralement collecté dans les phases de vision préliminaire et d'architecture.

Business Architecture

Les éléments de base de l'architecture métier sont destinés à capturer les modèles architecturaux relatifs aux opérations métier. Ceci en regardant spécifiquement les facteurs qui motivent l'entreprise, qui définissent son organisation et sa structuration, mais aussi les capacités fonctionnelles dont elle dispose.

Information Systems Architecture

Les éléments de base de l'architecture des systèmes d'information sont destinés à capturer les modèles architecturaux relatifs aux systèmes informatiques, en en documentant les applications et les données.

Technology Architecture

Les éléments de base de l'architecture technologique sont destinés à capturer les ressources technologiques acquises et utilisées pour implémenter et réaliser les solutions relatives aux systèmes d'information.

Architecture Realization

Les blocs de base de la réalisation d'architecture sont destinés à capturer les feuilles de route de changement montrant la transition entre les états d'architecture et les instructions de liaison qui sont utilisées pour orienter et gérer l'implémentation de l'architecture.

Le métamodèle de TOGAF est extensible : il propose un ensemble d'éléments de construction optionnels apportant des solutions pour exprimer des besoins d'architectures plus spécifiques. La Figure 10 présente brièvement les extensions actuellement proposées par TOGAF :

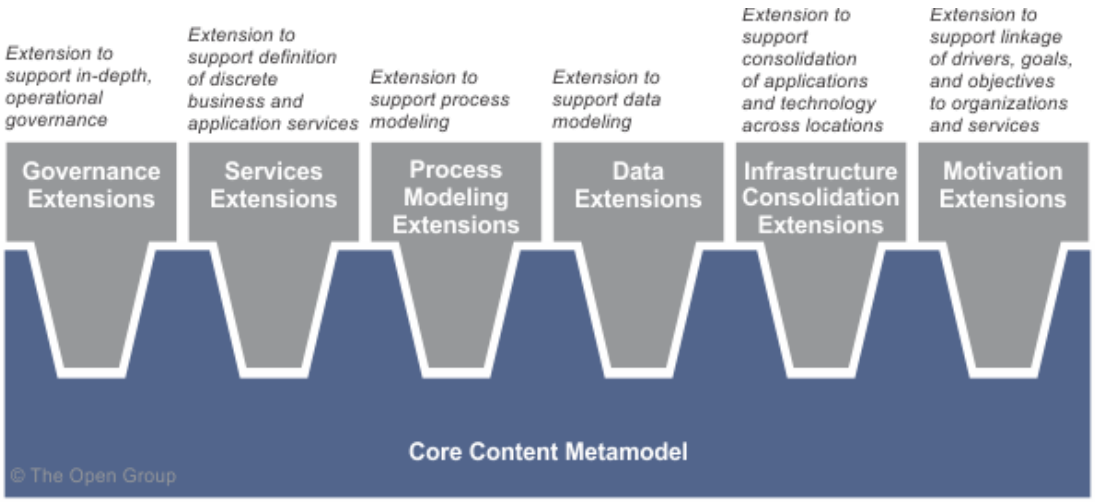

Figure 10 : TOGAF – Métamodèle de contenu de base et ses extensions [TOGAF 9.1]

Des informations plus détaillées sur le cadre de contenu de TOGAF et de ses extensions sont proposées dans la 2ème partie de ce mémoire.

Artefacts, bloc de construction et livrable

Le cadre de contenu d'architecture utilise trois catégories différentes pour classifier les résultats d'architecture produits durant l'exécution de l'ADM. Les trois catégories sont :

- Les **artefacts** (diagrammes, matrices, catalogues) et leurs relations avec le cycle ADM. Ils exposent une vue particulière de l'architecture et serviront de support de communication.

- Les **blocs de construction** (Building Blocks) sont les composants essentiels de l'architecture dont ils constituent l'ossature. Un Bloc de construction est un ensemble de fonctionnalités définies pour répondre à un besoin métier au sein de l'entreprise.

- Les **livrables**, sont des documents construits à partie des éléments précédents. Ces livrables sont validés formellement par les parties prenantes en sortie des différentes phases du cycle ADM. Ces livrables sont soit archivés à la fin du projet, soit déplacés vers le référentiel d'architecture en tant que modèle de référence, standard ou instantané du paysage d'architecture.

  La Figure 11 présente les trois catégories de résultats produits durant l'AE ainsi que leurs relations :

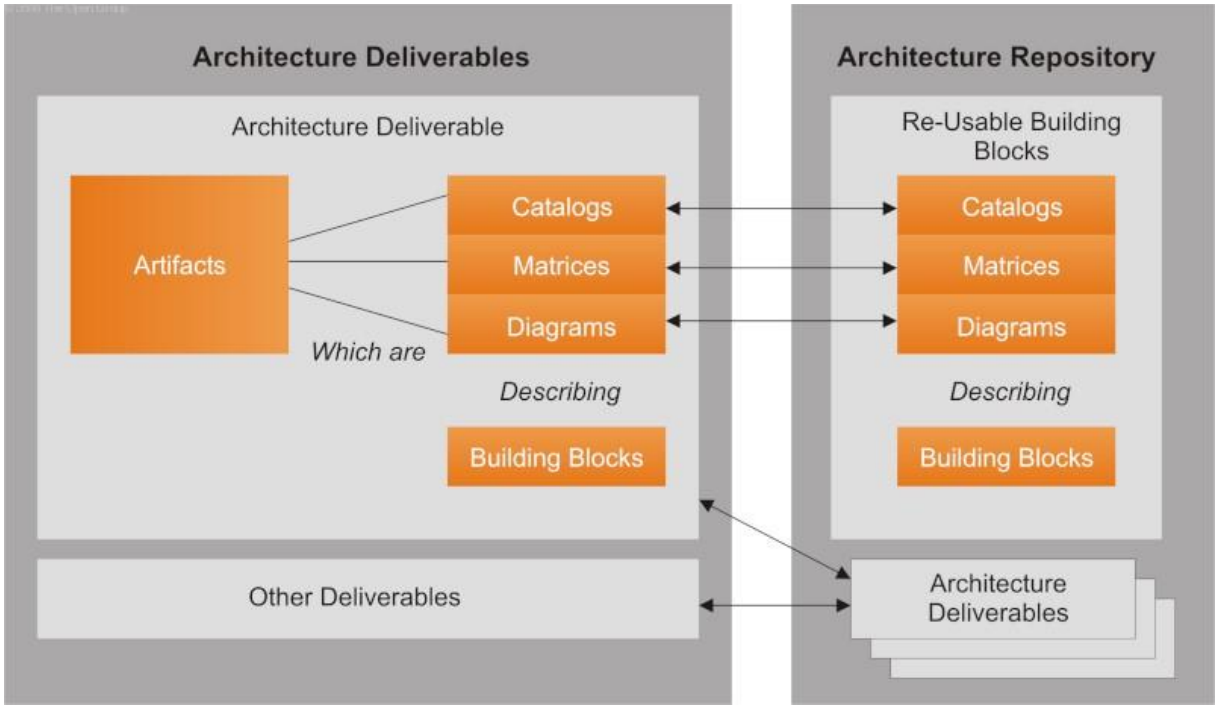

Figure 11 : TOGAF – Les constituants d'architecture [TOGAF9.1]

PARTIE V : Continuum d'entreprise (Enterpise continuum)

TOGAF recommande l'utilisation du continuum d'entreprise, ce dernier propose un plan d'organisation du référentiel d'entreprise ainsi que des méthodes de classification des artefacts d'architecture.

Le Continuum de l'Entreprise est un :

> « Espace de référence permettant de gérer les éléments nécessaires au bon fonctionnement de l'entreprise. Cet espace contient, entre autre, les continuums d'architecture dans leur évolution de Socle d'architecture en Architecture spécifique à l'organisation. »

[TOGAFGuideDePoche, 2010]

La Figure 12 présente les éléments constitutifs du continuum d'entreprise et leurs relations :

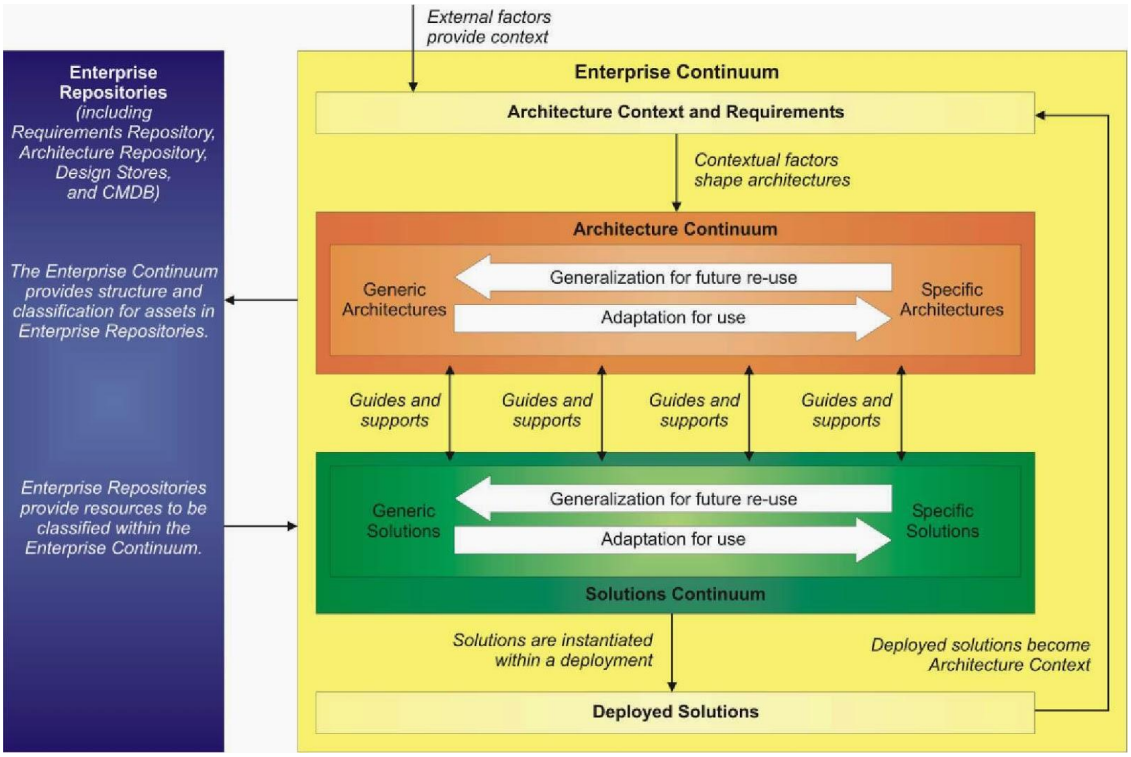

Figure 12 : TOGAF – Continuum d'entreprise [TOGAF9.1]

L'axe de structuration vertical du continuum d'entreprise laisse apparaître dans sa partie supérieure la représentation logique de l'architecture (Architecture Continuum), tandis que dans sa partie inférieure est représentée la réalisation physique de l'architecture (Solutions Continuum).

De plus, le continuum d'entreprise présente un axe de structuration horizontal selon lequel l'architecture « plus générique » (gauche) fait l'objet d'une adaptation pour en permettre l'usage, ceci en la déclinant en architecture « plus spécifique » (droite).

Ces deux axes permettent d'affiner l'architecture de « logique » à « physique » et de « plus générique » à « plus spécifique », à mesure de la progression depuis le problème initiale vers la solution finale.

Le terme « continuum » caractérise ce type de découpe utilisé qui partitionne les éléments du plus général au plus particulier : respectivement de l'architecture de fondation jusqu'aux architecture spécifiques. Le plan de classement suivant un niveau hiérarchique d'abstraction décroissant comporte quatre types d'éléments, tel que présenté dans la Figure 13 :

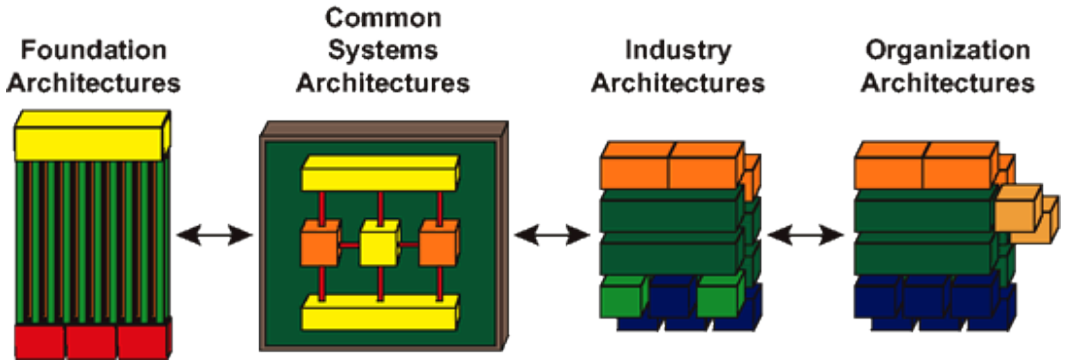

Figure 13 : TOGAF – Continuum d'architecture [TOGAF9.1]

Architectures de fondation

Socles d'architecture générique contenant des spécifications, des patterns d'architecture de haut niveau applicables à tous types d'entreprises. TOGAF donne un exemple de socle d'architecture : le TRM (voir TRM (Technical Reference Model)).

Systèmes communs

Sont des systèmes hautement réutilisables et dédiés à des services de nature transverse. (Ex : sécurité, réseau, communication, outil de gestion de workflow, …). TOGAF fournit un exemple de système commun : l'III-RM (voir L'IIIRM (Integrated Information Infrastructure Reference Model)).

Architectures spécifiques à un domaine

Sont aussi bien des modèles de données que des cadres de système d'information ou tout autre ensemble dédié à un domaine spécifique (Ex : ENERGISTICS[13] pour l'industrie pétrolière et gazière, ACORD[14] pour les assurances, …).

Architectures spécifiques à l'entreprise

Sont les éléments produits durant l'exécution des phases de l'ADM et que l'on désire capitaliser et potentiellement réutiliser. Ces éléments concernent aussi bien l'entreprise complète ou une partie de l'entreprise.

[13] http://www.energistics.org
[14] http://www.acord.org

PARTIE VI : Modèles de référence (Reference Models)

Dans cette partie TOGAF propose deux exemples détaillés du continuum d'architecture :

- Le TRM (Technical Reference Model)
- L'IIIRM (Integrated Information Infrastructure Reference Model)

TRM (Technical Reference Model)

Le TRM se présente comme une architecture de fondation au sein du continuum d'architecture. Il définit les constituants d'une infrastructure de système informatique en fournissant une terminologie, une structure et des règles d'interconnections entre les différents composants. La Figure 14 présente cette structure.

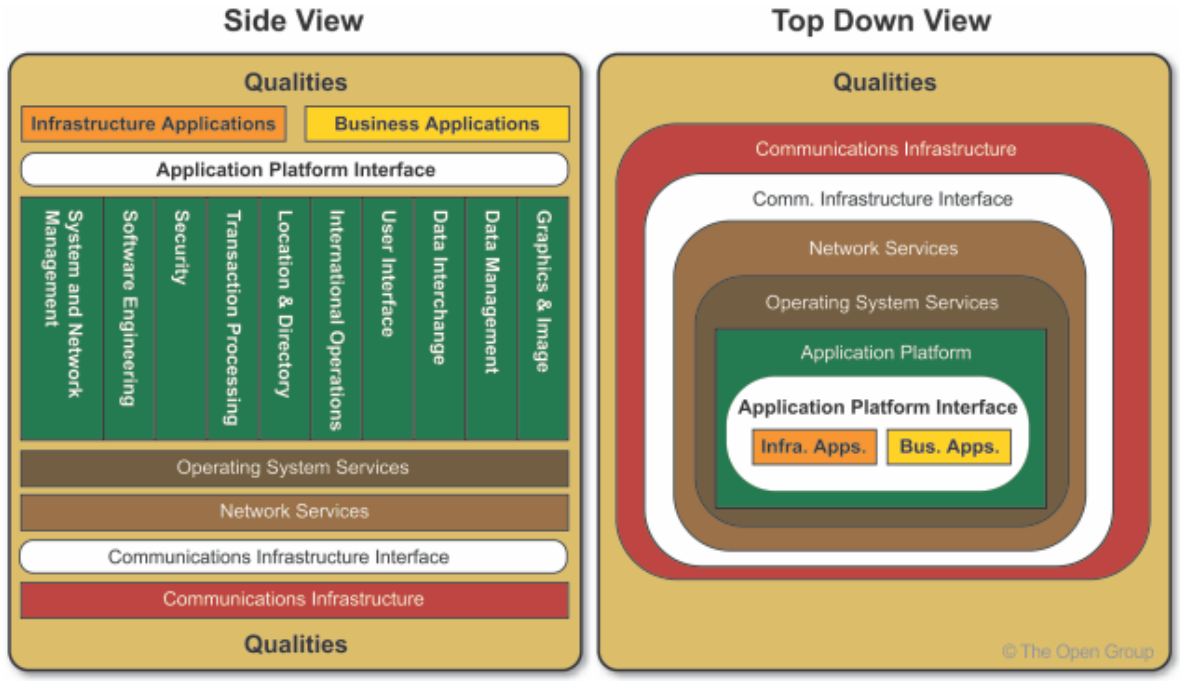

Figure 14 : TOGAF – Technical Reference Model (TRM) [TOGAF9.1]

Le TRM se structure en plusieurs niveaux, de l'infrastructure de communication jusqu'aux applications. Les applications s'appuient sur une interface dédiée, la plateforme de service offre une collection de services communs utilisés par l'ensemble des applications du système (ex : IHM, sécurité, …). Ces services communs sont construits sur deux couches de plus bas niveau qui sont la gestion du réseau et le système d'exploitation.

L'IIIRM (Integrated Information Infrastructure Reference Model)

Ce deuxième modèle de référence est un sous-ensemble du premier modèle de référence (TRM) focalisé sur les applications.

L'III-RM comprend les éléments clés permettant l'élaboration, la gestion et l'exploitation d'une infrastructure d'information intégrée. La Figure 15 en présente les éléments :

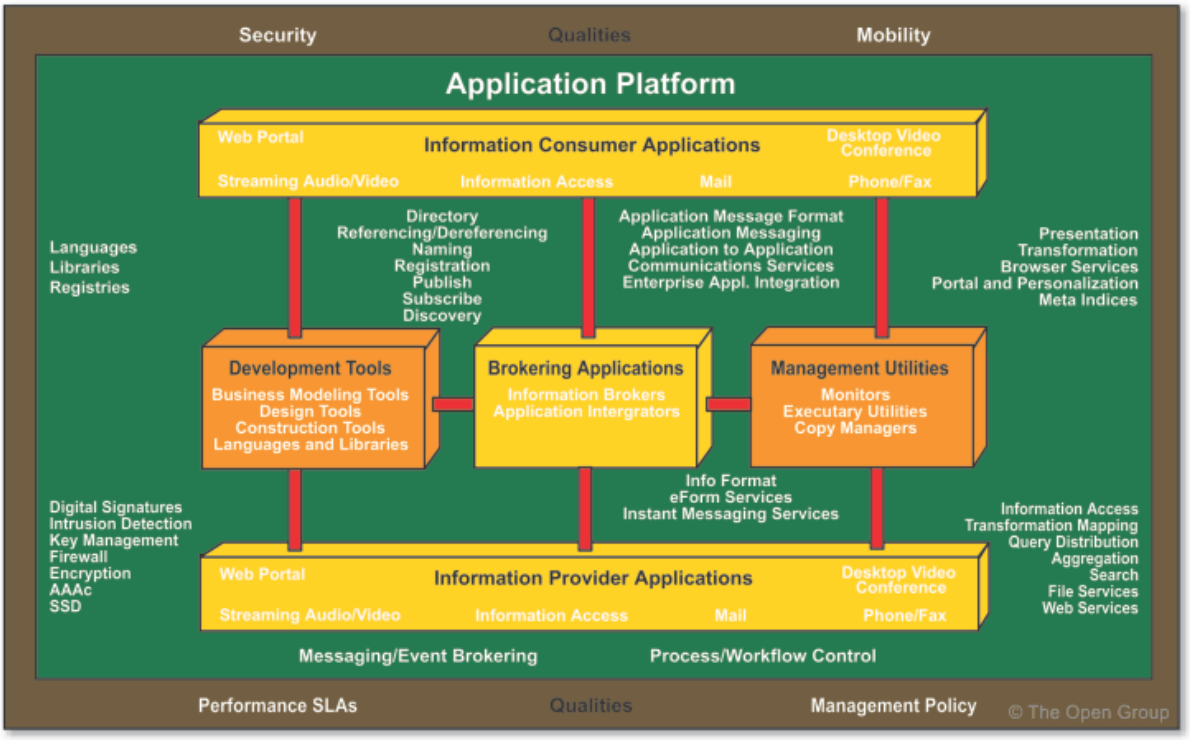

Figure 15 : TOGAF – Integrated Information Infrastructure Reference Model (III-RM) [TOGAF9.1]

PARTIE VII : Cadre de capacité (Architecture Capability Framework)

Le **Cadre de Capacité** permet de guider l'entreprise dans la mise en place de ce qui est nécessaire afin d'être en capacité de supporter la démarche d'architecture d'entreprise. Cette démarche nécessite donc de définir les rôles, responsabilités, processus, structures et compétences requises au sein de l'entreprise afin de rendre l'entreprise capable de réaliser l'Architecture d'Entreprise telle que définie par TOGAF.

« Le Cadre de Capacité est un ensemble de ressources, de recommandations, de modèles, d'informations de base, etc., aidant l'architecte à établir une pratique d'architecture au sein d'une organisation. »

[OGF, 2012]

« Une capacité est une aptitude offerte par une organisation, une personne ou un système. Les capacités sont exprimées en termes généraux, en utilisant des concepts d'un haut niveau d'abstraction. Elle nécessite pour sa mise en œuvre une combinaison d'organisations, de personnes, de processus et de technologies. »

[OGF, 2012]

La Figure 16 présente la structure générale du cadre de capacité :

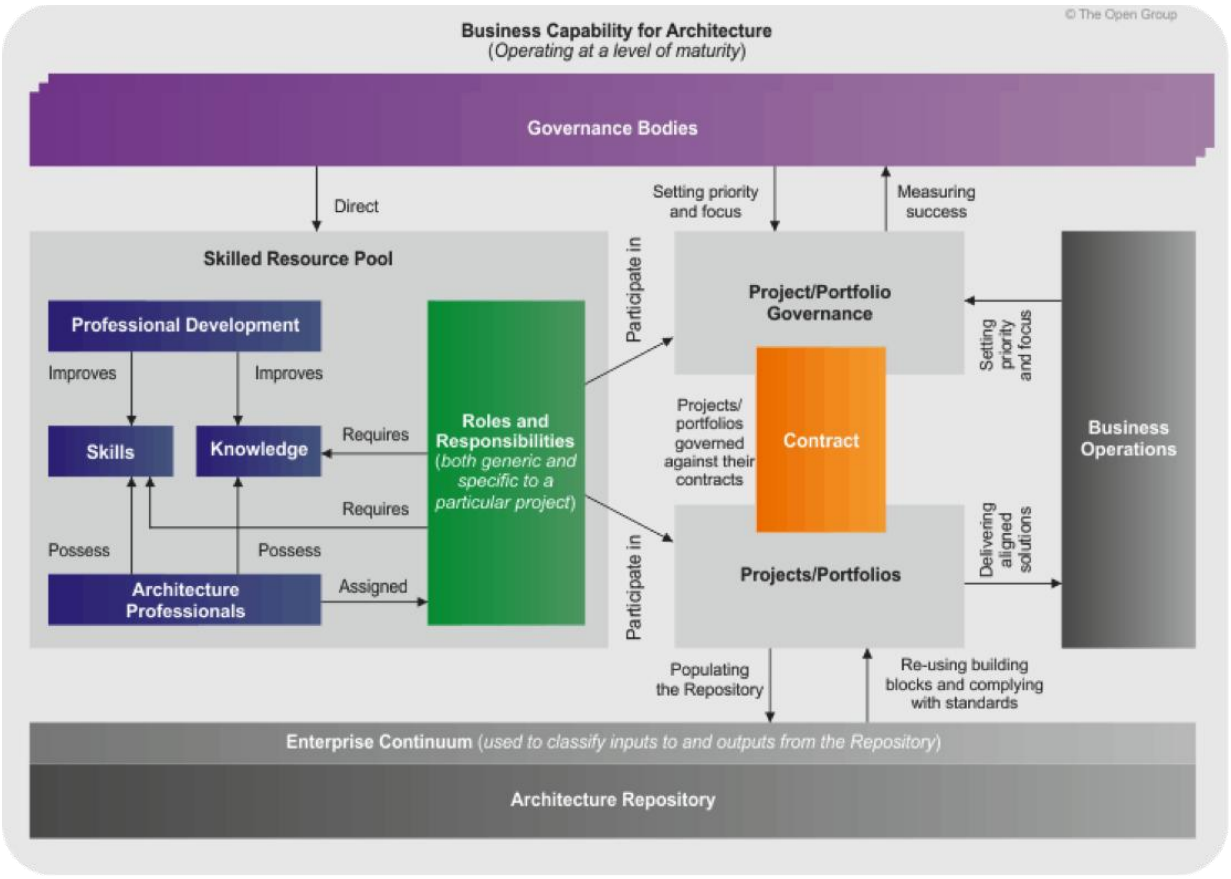

Figure 16 : TOGAF – Cadre de Capacité [TOGAF9.1]

En nous référant à la Figure 16 :

- L'entreprise doit définir les organes de gouvernance (Governance Bodies) qui seront responsables de prendre des décisions relatives aux priorités et de mesurer les succès engrangés au travers de la gouvernance des projets.

- Les organes de gouvernance sont également responsables du groupe de ressources qualifiées (Skilled Ressource Pool) au sein duquel sont identifiés les profils des professionnels de l'architecture (Architecture Professionals). Ces profils disposent de l'expertise (Skills) et de la connaissance nécessaire (Knowledges) afin de mener à bien les rôles et les responsabilités (Roles and Responsibilities) qui leur sont dévolus. Ces rôles concernent autant la participation à la gouvernance des projets que les projets eux-mêmes.

- L'organe de gouvernance des projets (Project/Portfolio Governance) planifie et organise la réalisation des projets en établissant les contrats de réalisation encadrant ces derniers. Ces contrats tiennent compte de la direction et des priorités soumises par les opérations métier.

- Les projets s'engagent à respecter les contrats d'architecture soumis par l'organe de gouvernance des projets. Dans le cadre de leur exécution, les projets (Projects) sont alimentés par les standards, les livrables, les artefacts et les blocs de construction réutilisables récupérés du dépôt d'architecture. Les résultats d'architecture produits par les projets sont consignés au sein du dépôt d'architecture. Les documents entrants/sortants du dépôt d'architecture sont classifiés selon les principes décrits par le continuum d'entreprise (enterprise continuum). Les projets mettent à la disposition du métier les solutions produites.

1.2.4 TOGAF et les styles d'architecture logiciels

TOGAF est un cadre générique, flexible et extensible qui peut être facilement adapté à un certain nombre de styles d'architecture, et de cette façon être utilisé dans une grande variété d'environnements.

Les styles architecturaux diffèrent en fonction de leur préoccupation, de la forme, des techniques, des matériaux, du sujet et du temps. Certains styles peuvent être considérés comme à la mode, d'autres axés sur des aspects particuliers de l'architecture d'entreprise.

On peut s'attendre à ce que le paysage d'architecture d'une organisation contienne des travaux d'architecture développés dans de nombreux styles d'architecture. TOGAF veille à ce que les besoins de chaque partie prenante soient abordés de manière appropriée dans le contexte d'autres parties prenantes et de l'architecture de référence.

Depuis sa version 9, TOGAF est accompagné d'une nouvelle partie intitulée « PARTIE III : Recommandations et Techniques pour l'ADM ». Parmi les informations proposées dans cette partie on trouve deux documents de références présentant l'intégration de deux styles architecturaux qui sont :

- Architecture de sécurité et ADM
  (Voir [TOGAF9.1], section 21. Architecture de sécurité et ADM)

- Utilisation de TOGAF pour définir et gouverner les SOAs
  (Voir [TOGAF9.1], section 22. Utilisation de TOGAF pour définir et gouverner les SOAs)

Au fil du temps, de nouveaux styles architecturaux devraient se poser pour résoudre les principaux problèmes auxquels sont confrontés les praticiens. Certains styles seront transitoires, certains survivront dans un créneau, et certains fusionneront dans le courant principal. Les forums et les groupes de travail Open Group existent pour relever les défis auxquels l'industrie est confrontée. Ces groupes de travail produisent un large éventail de matériels utiles aux praticiens intéressés d'adapter TOGAF à un style architectural particulier.

Pour consulter les matériaux actuels, y compris les livres blancs et les normes applicables voir www.opengroup.org/ Togaf_docs.

1.3 DDD – Domaine Driven Design

Ce chapitre aborde le style d'architecture DDD et la façon dont cette approche de conception logicielle s'organise. Cette présentation permet d'en apercevoir l'essentiel afin de nous permettre d'identifier les éléments pertinents à considérer dans le cadre de l'intégration du style d'architecture DDD au sein de TOGAF.

1.3.1 Présentation

La conception dirigée par le domaine (Domain-Driven Design ou DDD) est une approche de conception logicielle qui vise à accorder en premier lieu de l'importance au domaine métier.

Les trois principes au cœur de la démarche DDD :

- L'attention est portée sur la logique et le cœur du domaine métier.
- Les conceptions complexes se basent sur les modèles du domaine.
- Les experts du domaine collaborent en permanence, afin d'améliorer le modèle applicatif et de résoudre tous les problèmes liés au domaine.

Le style d'architecture DDD propose une approche permettant de faciliter la réalisation de projets informatiques s'exécutant et interagissant au sein de domaines métiers complexes. La vision orientée modèle accorde une grande importance à la compréhension et à la communication du domaine métier. Ceci afin d'en réduire la complexité et de la sorte faciliter la compréhension et la maintenance des modèles métiers.

Eric Evans, l'initiateur de cette approche, justifie sa nécessité en mettant en avant le décalage entre l'importance de la logique métier présente dans les logiciels (ce qui forme leur réelle plus-value) et le positionnement du modèle métier trop souvent « effacé » au profit d'autres parties du développement plus techniques comme par exemple l'interface graphique, la persistance des données, etc.

Afin de soutenir sa démarche, DDD défini un ensemble de concepts et de recommandations permettant sa mise en œuvre. La Figure 17 en est une des représentations les plus courantes. Cette figure présente l'ensemble des éléments proposés par la démarche et en organise les relations.

Parmi les éléments présentés à la Figure 17 se trouvent plusieurs sous-ensembles de recommandations et de concepts répondants de différentes façons aux besoins de la démarche. Ces sous-ensembles sont :

- **Les techniques de compréhension et de communication du domaine**, sont un ensemble de techniques encadrant l'étude d'un domaine métier. Parmi ces techniques se trouvent l'usage d'un langage omniprésent et un ensemble de techniques de distillation du domaine métier qui permettent d'en identifier et classifier les éléments importants.

- **Les techniques de préservation de l'intégrité du modèle**, sont un ensemble de techniques permettant de garantir la cohérence et l'intégrité du modèle tout au long de son cycle de vie. Parmi ces techniques, la notion de contexte borné et les stratégies de préservation de l'intégrité des domaines apportent des solutions pertinentes permettant de maitriser les évolutions des modèles et de formaliser leurs relations.

- **La conception dirigée par le modèle**, propose un ensemble de concepts permettant la mise en œuvre des éléments nécessaires permettant de soutenir une approche de conception dirigée par le modèle.

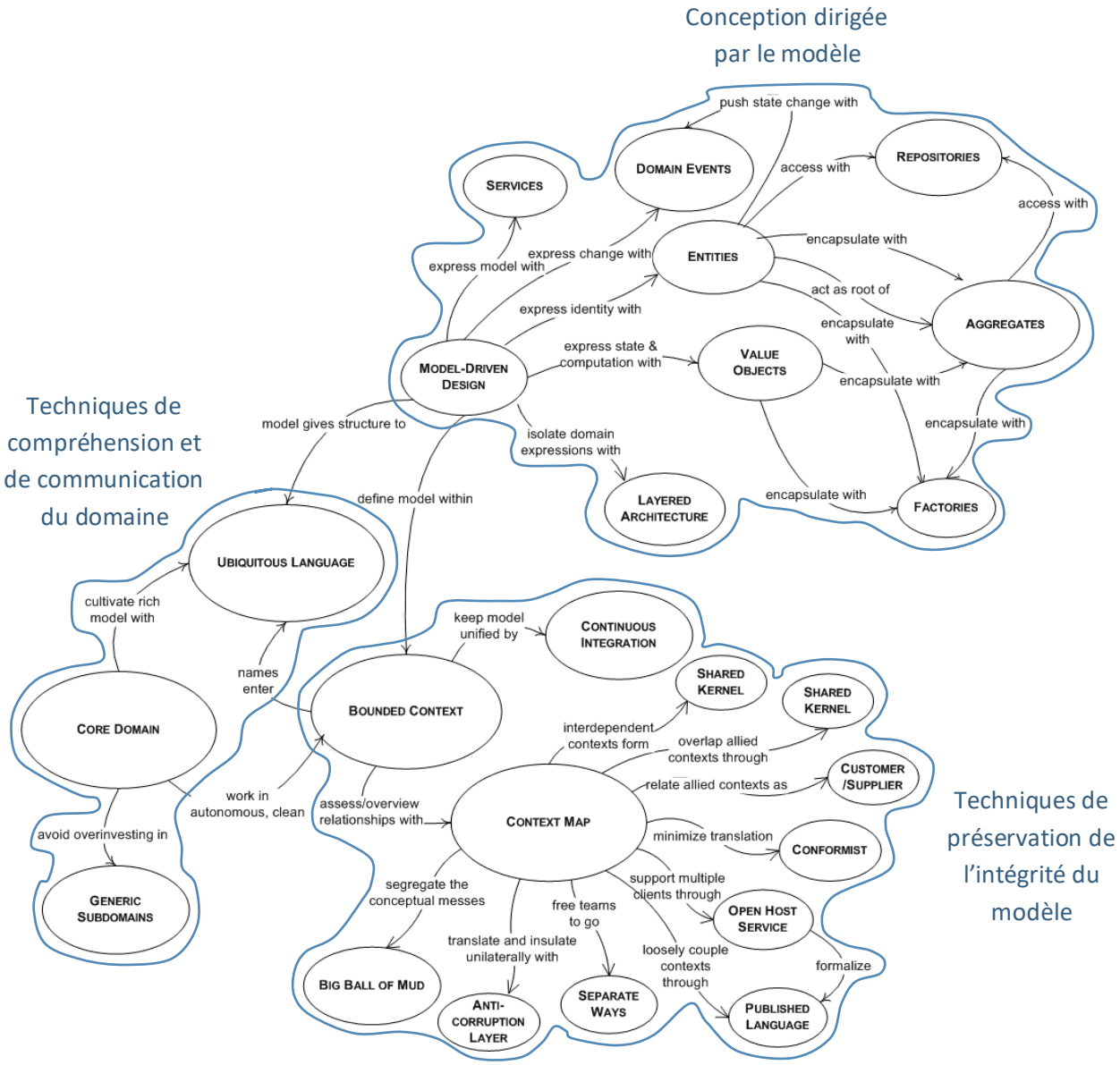

Figure 17 : DDD – Vue d'ensemble [DDD, 2004]

1.3.2 Historique

DDD doit son origine aux travaux menés par Eric Evans. Depuis le début des années 90, Eric Evans a travaillé sur de nombreux projets informatiques. Dans ce contexte, il a contribué au développement de grands systèmes métiers, ceci avec de nombreuses approches et résultats différents. DDD est une synthèse de cette expérience. Il formalise un style d'architecture, que des équipes ont utilisé avec succès pour aligner des systèmes logiciels complexes sur les besoins métiers, et garder des projets agiles au sein de systèmes grandissants.

La première et principale parution de DDD est un ouvrage de Eric Evans, intitulé « Domaine Driven-Design : Tackling Complexity in the Heart of Software », en 2004.

Suite au travail d'Eric Evans, de nombreux ouvrages reprenant et appliquant ses idées ont vu le jour. Bien que ces ouvrages ne redéfinissent pas la méthode, ils ont permis de contribuer à l'adoption de plus en plus importante de DDD au sein de la pratique de conception logicielle.

Récemment, en 2015, Eric Evans à publié « DDD Reference » qui se présente comme un bref résumé de toutes les définitions et patterns abordés dans le livre sorti en 2004. Celui-ci apparait aussi comme un complément de l'ouvrage précédent dans la mesure où il documente deux techniques et un concept supplémentaires ne se trouvant pas dans le livre original. Cet ajout reflète l'évolution de la pensée d'Eric Evans et de ce qu'il considère aujourd'hui comme faisant partie de sa compréhension de DDD. Ces trois ajouts sont respectivement :

- **Partenariat** (PartnerShip) et **Grande boule de boue**[15] (BigBall of Mud), deux nouvelles stratégies se joignant aux techniques de préservation de l'intégrité du modèle.

- **Evénement du domaine** (Domain Event), un nouveau concept qui prend place parmi les concepts soutenant la conception logicielle guidée par le modèle.

L'ajout de ces trois patterns constitue la seule évolution significative de DDD depuis sa formalisation et jusqu'à ce jour.

---

[15] Le terme a été popularisé par Brian Foote et Joseph Yoder dans leur article de 1999 « Big Ball of Mud », avec la définition suivante : « Une grande boule de boue est une vaste jungle de code mal structuré, programmé en spaghetti, peu soigné et souvent rafistolé. Ces systèmes témoignent clairement des traces d'une expansion incontrôlée, ainsi que de fréquentes réparations opportunes et improvisées. Les informations sont partagées sans distinction parmi les éléments distants du système, souvent jusqu'au point où presque toutes les informations importantes deviennent globales ou dupliquées. La structure du système dans son ensemble n'a peut-être jamais été bien définie. Si elle l'a été, le système a été tellement détérioré qu'il est impossible de la reconnaître. Les programmeurs avec un brin de sensibilité architecturale fuient ces bourbiers. Seuls ceux qui ne sont pas préoccupés par l'architecture, et qui, peut-être, ne sont pas dérangés par l'inertie d'une corvée quotidienne consistant à coller des rustines sur ces digues défaillantes, sont heureux de travailler sur de tels systèmes. »

1.3.3 Principes

Comprendre et communiquer le domaine

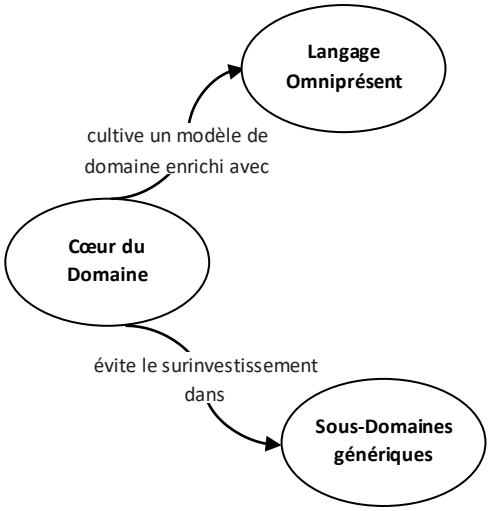

Figure 18 : DDD – Comprendre et communiquer le domaine [DDD, 2004]

Comprendre et communiquer le domaine métier est un aspect très important dans l'approche DDD. Cette approche domaine-centrique soutient l'idée suivante : ce n'est qu'une fois le domaine métier compris par tous les acteurs impliqués dans la démarche de conception logicielle que le logiciel pourra devenir le reflet du métier et lui apporter une réelle plus-value.

Très orientée vers la compréhension du domaine et sa communication, DDD fourni une série de bonnes pratiques et de nombreux exemples afin de guider les efforts d'analyse et de modélisation d'un domaine métier. Cette approche positionne tous les acteurs impliqués dans la conception logicielle autour du domaine métier et de ses modèles.

DDD rappelle que le domaine métier est avant tout connu des spécialistes (du domaine) et que ce sont eux qu'il faut rencontrer pour en comprendre toutes les subtilités. Cependant, DDD ne fait pas de la compréhension du domaine métier l'apanage de l'analyste et de l'expert. L'approche recommande d'impliquer tous les acteurs de la conception logicielle dès les travaux de modélisation du domaine afin de cultiver une vision et une compréhension commune des modèles. L'implication et la bonne compréhension du domaine métier par tous les acteurs doit, selon DDD, permettre d'adapter directement les modèles afin que ces derniers puissent être utilisés en l'état par les différents acteurs. Pour ce cela, DDD préconise de faire évoluer les modèles selon l'émergence et le raffinement des différentes contraintes durant tout le processus de production logiciel (contraintes de performances, contraintes de sécurité, …). Rendre les modèles utilisables par tous les acteurs métier évitera que ces derniers ne doivent construire leur propre modèle, car ces abstractions supplémentaires du domaine risquent de créer une fracture entre le domaine et le logiciel.

Le langage omniprésent (Ubiquitous Language)

Afin d'aider à la bonne compréhension du domaine, DDD propose la mise en place du langage omniprésent. Cette pratique recommande la mise en œuvre d'un langage commun au sein d'un contexte métier donné. Le problème auquel cette pratique tente de répondre est le frein à la communication que provoque l'utilisation de plusieurs langages spécialisés au sujet d'un même domaine (langage des experts, langage de développeurs, …). Il est apparaît donc primordial d'élaborer un langage commun et compréhensible de tous. Ce vocabulaire commun doit permettre :

- d'identifier les concepts clés qui définissent le domaine et guident la conception
- d'identifier les expressions utilisées dans le domaine
- de chercher et résoudre les ambiguïtés et les inconnues

La création d'un tel langage ne se fait pas en une seule fois. Ce dernier est élaboré progressivement au fil des discussions avec les experts du domaine.

Le Cœur du domaine (Core Domain)

Partant du constat suivant : « Dans la conception d'un grand système, il y a tellement de composants contributifs, tous compliqués et tous absolument nécessaires au succès, que l'essence du modèle de domaine, l'atout réel de l'entreprise, peut être obscurci et négligé. » [DDD, 2004, page 400]

DDD défini une approche systématique du processus de fragmentation du domaine métier. Cette fragmentation, ici nommée distillation, a pour objectif d'extraire l'essence du domaine métier et de distinguer le domaine principal des domaines génériques supportant sa réalisation.

L'identification du domaine principal est motivée par le désir d'identifier la partie du domaine métier qui possède une valeur particulière et qui distingue le logiciel de l'entreprise des autres systèmes.

L'adoption d'une démarche de distillation du domaine métier permet de :

- Aider les membres de l'équipe à comprendre la conception globale du système.
- Faciliter la communication en identifiant un modèle central de taille gérable.
- Guider les refactorisations.
- Concentrer le travail sur les parties du modèle ayant le plus de valeur pour l'entreprise.
- Guider la sous-traitance, l'utilisation de produits du marché et les décisions concernant les affectations.

La Figure 19 présente les éléments encadrant la distillation du domaine métier :

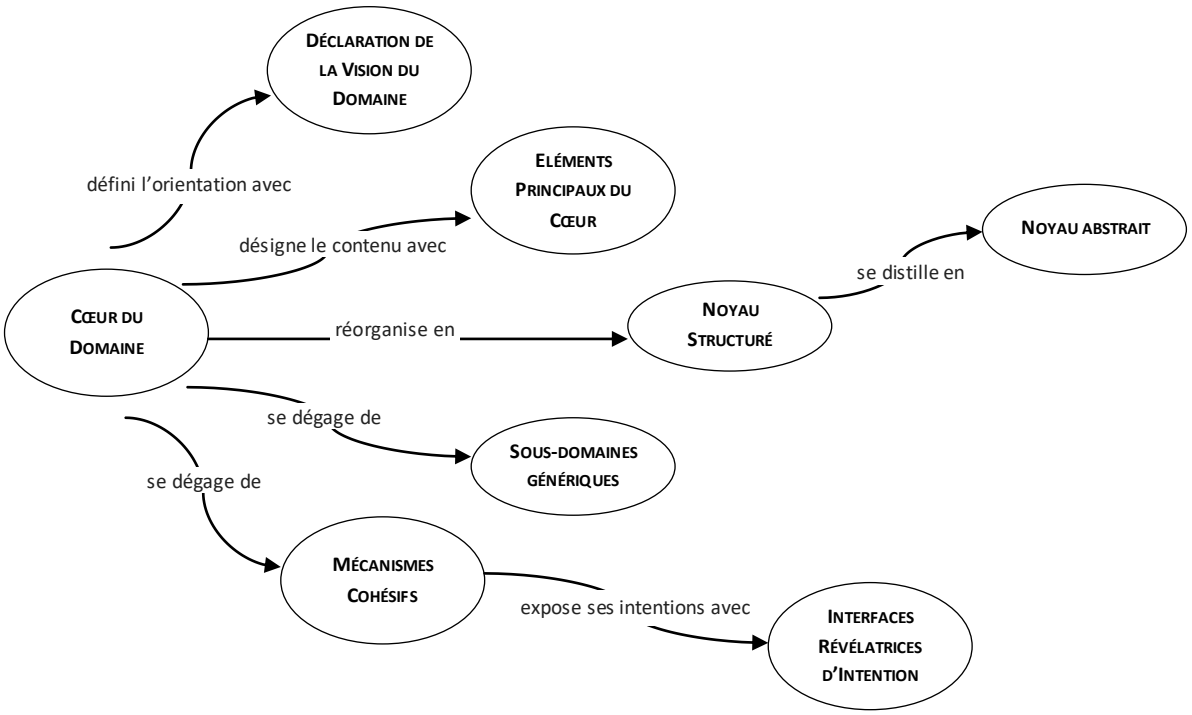

Figure 19 : DDD – Distillation du domaine métier [DDD, 2004]

Les différentes techniques de distillation qui se concentrent sur l'organisation du Domaine peuvent être appliquées dans presque n'importe quel ordre. Néanmoins, il existe un degré d'impact différent dans la façon dont elles modifient la conception du domaine. Ces techniques de distillation sont :

Déclaration de la vision du domaine

La déclaration de la vision du domaine capture et communique les concepts de base et les objectifs et valeurs attendues du domaine métier.

Les éléments principaux du Cœur

Les éléments principaux du cœur mettent en avant les éléments majeurs afin de faire le lien avec la vision du domaine. Cette perspective met en avant le cœur du domaine, aide à la communication (donne une vision des concepts essentiels) et guide la décision (ex : concentrer le travail sur les parties du modèle ayant le plus de valeur pour l'entreprise).

Les sous-domaines génériques

Les sous-domaines génériques isolent les parties du modèle qui ajoutent de la complexité sans pour autant capturer ni communiquer de savoir particulier à l'entreprise. L'allégement du cœur du domaine de ces concepts génériques en permet une meilleure compréhension.

Le noyau structuré

Le noyau structuré est le résultat de l'identification et de la classification de sous-domaines cohérents au sein du cœur du domaine métier.

Le noyau abstrait

Le noyau abstrait est le résultat de l'identification et de l'abstraction de concepts les plus fondamentaux du modèle. Cette technique s'applique lorsqu'une découpe horizontale permet d'identifier des interfaces polymorphiques représentant des concepts fondamentaux du domaine. Il est alors recommandé par DDD de créer l'abstraction nécessaire pour isoler ces concepts, ce qui permettra de diminuer le couplage entre les modules du cœur du domaine.

Les mécanismes cohésifs

Les mécanismes cohésifs isolent du cœur du domaine de certains algorithmes très complexes et rendant difficile sa compréhension. Cette technique permet de séparer le « quoi » du « comment ». La conception du cœur du domaine sera, dès lors, plus simple à comprendre et à utiliser.

Les Interfaces révélatrices d'intention

Les interfaces révélatrices d'intention permettent d'exposer des mécanismes cohésifs au moyen d'affirmations conceptuellement cohérentes et de fonctions exemptes d'effets de bord. Ces mécanismes alliés à une bonne conception, permettent au cœur du domaine d'utiliser des assertions significatives plutôt que des fonctions obscures. L'objectif de cette démarche est de construire un langage permettant d'exprimer les scénarios d'application les plus importants de manière souple et concise.

Techniques de préservation de l'intégrité du modèle

DDD fourni un ensemble de techniques pour maintenir l'intégrité du modèle. Les domaines métiers de taille réduite et peu complexes peuvent s'exprimer sous la forme d'un modèle unifié dont il ne sera pas difficile de maintenir l'intégrité tout au long du processus de conception. Si cela est vrai pour des domaines métiers peu complexes, il n'en va pas de même pour les grandes entreprises. Ces dernières disposent généralement de domaines métiers très complexes et devant le plus souvent interagir avec des applications externes, possédant elles-mêmes leur propre domaine métier (et donc leur propre modèle pas nécessairement compatible avec celui de l'entreprise). Il convient dès lors de proposer une série de recommandations permettant de garantir l'intégrité des modèles de l'entreprise. Mais également de faciliter la segmentation du domaine de l'entreprise afin de pouvoir y organiser le travail collaboratif.

La Figure 20 présente ces techniques et les relations qui existent entre elles :

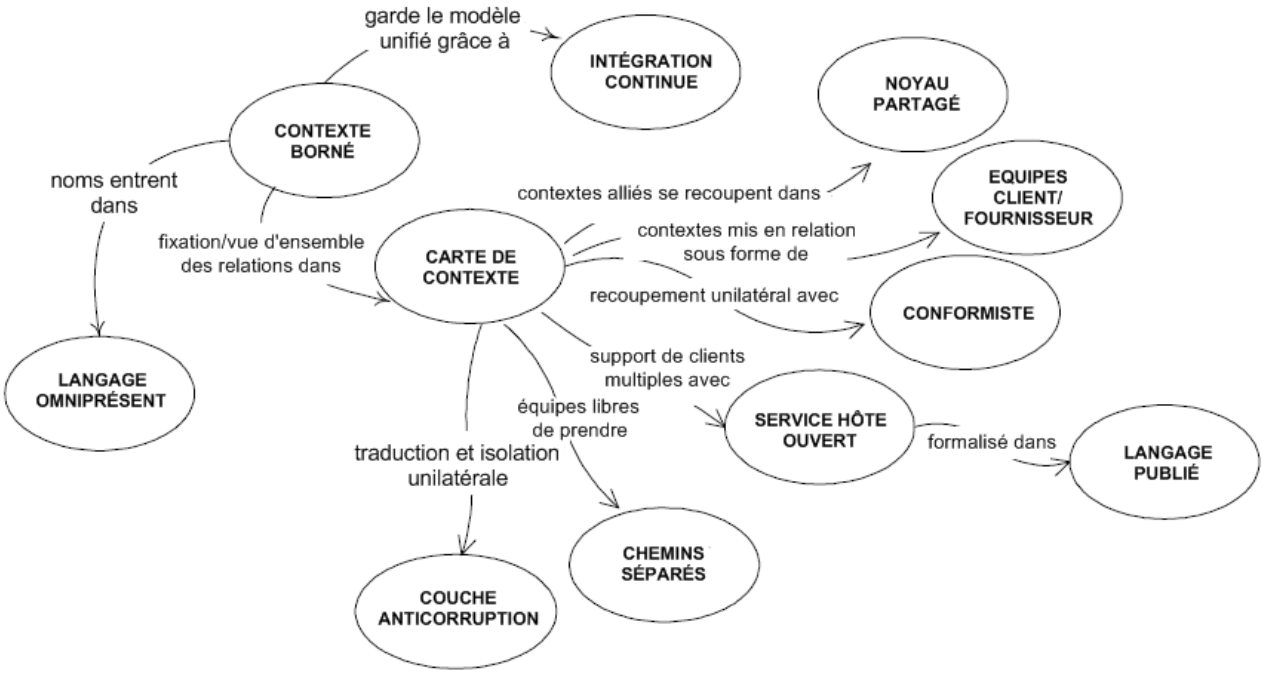

Figure 20 : DDD – Techniques de préservation de l'intégrité du domaine métier [DDD, 2004]

Les contextes bornés (Bounded Context)

La définition de contextes bornés est une des étapes les plus importantes de l'approche DDD. Cette technique s'applique lorsque le domaine métier est particulièrement vaste et que l'unification de son modèle représente un idéal difficile à atteindre. Afin de répondre à cette problématique, DDD recommande la définition de contextes bornés. Ces derniers segmentent le modèle du domaine en sous-modèle unifiés distincts. Les contextes bornés ainsi créés fournissent les cadres logiques à l'intérieur desquels les modèles peuvent évoluer.

Ceci a pour avantage de faciliter la maintenance des modèles et de conduire à une meilleure indépendance des équipes de conception dans le cadre de projet complexe. Idéalement, un modèle doit être assez petit pour être assigné à une seule équipe.

L'intégration continue (Continuous Integration)

DDD recommande l'adoption de l'intégration continue afin de garantir l'intégrité des contextes bornés définis. La pratique vise à définir et maintenir une famille complète de tests (unitaire, fonctionnels, …) et un processus de mise en commun régulier du code produit par une ou plusieurs équipes opérant au sein d'un contexte borné. De cette façon, toute incohérence du modèle peut être repérée dans l'implémentation et prise en charge très tôt.

La carte de contexte (Context Map)

Une carte de contexte documente les différents contextes bornés et leurs relations. La réalisation de cette documentation permet de fournir une vision d'ensemble et de faciliter l'intégration des différents contextes bornés entre eux. Sur la carte de contexte, chaque contexte borné possède un nom qui fait partie de langage omniprésent.

DDD défini différentes stratégies pour organiser la relation de deux contextes bornés. Celles-ci peuvent être classées, de 3 façons différentes, en fonction de la relation entre contextes.

- **Haut niveau d'interaction entre contexte** : Des objets sont échangés de façon uni ou bi directionnelle entre deux contextes bornés. L'application de ces stratégies nécessite une coopération entre équipes.

Noyau partagé

Permet à deux contextes bornés (deux équipes) de partager un sous-ensemble de concepts. Chaque équipe dispose de son contexte borné dans lequel se trouve une partie du modèle qui est commune aux deux équipes et pour laquelle les équipes doivent se coordonner. Les modifications du domaine commun doivent toujours impliquer les deux équipes.

- Permet d'éviter une partie de l'intégration du code de l'autre équipe. Puisque le noyau est partagé, il n'y aura pas de traduction nécessaire des concepts du noyau entre les deux équipes.
- Permet d'éviter les doublons de concepts tout en gardant une partie des contextes séparés.

Des tests communs doivent être mis en ouvre afin de garantir l'intégrité du modèle partagé.

Equipes Client / Fournisseur

Permet à deux contextes bornés (deux équipes) de partager un sous-ensemble de concepts, ceci en délégant la totalité de la gestion du domaine commun à l'une des deux équipes. Cette technique est une alternative lorsque le Noyau Partagé ne peut être appliqué.

Une équipe détient la partie du modèle visé par la dépendance fonctionnelle, elle se positionne en tant que fournisseur par rapport à la deuxième équipe. La deuxième équipe considère le domaine commun comme intégré à son contexte borné, elle se positionne en tant que client de l'équipe fournisseur. L'équipe cliente doit énoncer ses besoins à l'équipe fournisseur qui doit y répondre avec des plans.

- Permet d'éviter les doublons de concepts tout en gardant une partie des contextes séparés.

Des tests de non régression doivent être mis en œuvre y compris par l'équipe cliente afin de garantir l'intégrité du modèle récupéré.

Conformisme

Permet à un contexte borné de consommer des objets issus d'un autre contexte borné. Le système tiers est le seul gestionnaire de son modèle. Cette technique diffère du Client / Fournisseur dans le sens où le client n'a aucune influence sur le fournisseur et doit donc se conformer aux modèles fournis par ce dernier.

Typiquement, cette solution est préconisée quand les données externes proviennent d'applications tierces ou au sein d'une équipe lorsque le management nécessaire n'est pas adéquat pour stimuler la coopération entre équipes. Le client n'a pas le choix de l'interface et il est compliqué d'obliger l'application tierce à adapter son interface selon les besoins du client.

- **Contexte fortement indépendant :** Des objets sont échangés de façon uni directionnelle ou des objets sont isolés des autres objets, l'application de ces stratégies ne nécessite aucune coopération entre équipes.

Chemin séparé

Permet de diviser un contexte métier en plusieurs sous contextes bornés. Un contexte borné séparé encapsule une partie du modèle métier et n'entretient aucune relation avec les éléments du domaine qui lui sont extérieurs. Cette technique conduit à un découplage complet des contextes bornés séparés vis-à-vis des autres contextes. Les contextes bornés séparés peuvent évoluer et être implémenté de façon autonome. Contrairement aux techniques précédemment citées qui nécessitent des efforts d'intégration parfois très important et une concertation des équipes concernant l'implémentation du modèle commun.

- Permet de rendre indépendante plusieurs parties de l'application d'entreprise qui n'ont pas d'interaction fonctionnelle spécifique.
- Permet aux contextes bornés séparés d'évoluer séparément.

Couche anti-corruption

Permet d'adapter les appels vers d'autres contextes en introduisant une couche de traduction des concepts métiers. Si un contexte doit s'interfacer avec plusieurs contextes alors il y aura autant de couches anticorruption que d'interfaces entre contextes.

- **Intégration de système historique / externe :** Des objets sont échangés de façon uni directionnelle, l'application de cette stratégie ne nécessite aucune coopération entre équipes.

Service hôte ouvert

Permet d'adapter les appels vers d'autres contextes en introduisant une couche de traduction des concepts métiers. Ceci est similaire à l'approche préconisée par la couche anticorruption à la différence que cette technique préconise de n'utiliser qu'une seule couche de traduction même si plusieurs contextes doivent s'interfacer avec le contexte en question.

- La logique de transformation doit être commune à tous les contextes clients.
- Tous les clients seront alors concernés dans le cas d'un changement de l'interface commune.
- L'intérêt de cette technique est d'éviter de dupliquer une logique de transformation pour plusieurs couches en particulier si elle est semblable d'un contexte client à l'autre.

Big ball of mud

Permet de dessiner une limite autour de tout le désordre pouvant exister dans un domaine. Le désordre ainsi encapsulé est isolé des autres modèles du domaine.

Cette technique est assez pratique dans certaines situations, généralement lorsque les modèles sont mélangés et que leur limites sont incohérentes. Des limites de contexte bien définies n'émergent que de choix intellectuels. Lorsque ce facteur est absent ou disparait, de multiples systèmes conceptuels se combinent, rendant les définitions et les règles ambiguës ou contradictoires.

La grande boule de boue est en fait assez pratique pour ces situations mais il empêche presque complètement le gain de subtilité et de précision nécessaires aux modèles.

Partenariat

Cette stratégie est applicable lorsque deux équipes dans deux contextes différents voient leur réussite ou leur échec intimement liés.

Il arrive en effet qu'une mauvaise coordination de sous-systèmes mutuellement dépendants et issus de contextes distincts entraîne un échec de la livraison des deux projets. Une caractéristique clé manquante d'un système peut rendre l'autre système non livrable. Les interfaces qui ne correspondent pas pourraient provoquer l'échec de l'intégration, ou bien encore une interface mutuellement convenue pourrait s'avérer si difficile à utiliser qu'elle ralentirait le développement du système client, ou si difficile à mettre en œuvre qu'elle ralentirait le développement du système serveur.

Lorsque l'échec du développement dans l'un ou l'autre des deux contextes entraîne un échec de la livraison pour les deux contextes, DDD recommande de forger un partenariat entre les équipes chargées de ces deux contextes interdépendants. La stratégie de partenariat permet de mettre en place un processus de planification coordonné du développement et de la gestion conjointe de l'intégration entre ces sous-systèmes.

Les équipes doivent coopérer sur l'évolution de leurs interfaces pour répondre aux besoins de développement des deux systèmes. Des processus clairs sont nécessaires pour régir l'intégration. Par exemple, une suite de test peut être définie afin de démontrer que l'interface développée réponde aux attentes du système client. Ces tests peuvent être exécutés dans le cadre d'une intégration continue sur le système serveur.

Conception dirigée par le modèle

Les techniques présentées jusqu'à présent étaient concentrées sur le processus de modélisation en vue de créer un bon modèle métier. Les techniques suivantes abordent la phase de conception logicielle, c.-à-d. la transition du modèle vers le code.

La représentation suivante présente ces techniques et les relations qui existent entre elles :

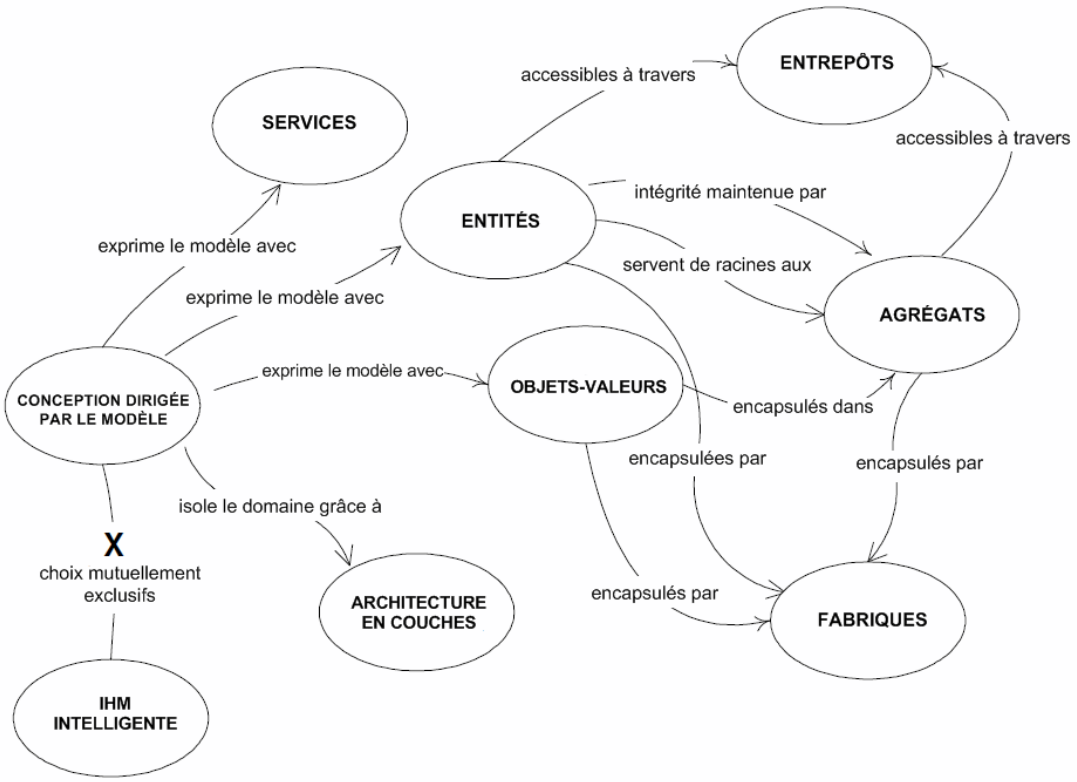

Figure 21 : DDD – Conception Dirigée par le modèle [DDD, 2004]

DDD recommande d'adopter un mode de conception dirigée par le modèle. Cette approche se concentre sur le résultat de la modélisation, son objectif est de lier étroitement la modélisation du domaine et la conception logicielle. La modélisation dirigée par le modèle recommande de concevoir le logiciel de telle sorte qu'il reflète le modèle du domaine de manière très littérale, de cette façon le code devient l'expression du modèle.

Cette approche nécessite une implication des développeurs très tôt dans le processus de modélisation. Ceci permet d'assurer que tout le monde comprenne les subtilités du modèle et de vérifier son implémentatibilité en code. De cette façon, le modèle peut si besoin être adapté pour faciliter son implémentation en tenant compte des limites introduites par les outils de développement.

Cette approche nécessite également une implication de l'analyste dans le processus de conception logicielle. Ceci afin de lui permettre d'être informé des préoccupations liées aux limites introduites par le développement et ainsi, de recueillir les refactorisations du code (via émergence ou évolution des exigences) qui conduiront la plupart du temps à un changement du modèle.

Si tel n'est pas le cas, il existe un risque que les développeurs s'écartent du modèle de base et produisent leurs propres modèles. A terme, cette rupture entre modèles aura pour conséquence que le modèle logiciel cessera d'exprimer le modèle original et perdra dès lors sa valeur fondamentale.

DDD préconise un certain nombre de techniques en ce qui concerne la conception du domaine (voir Figure 21). Toutefois, elle n'impose pas de les utiliser absolument. Comme dans la plupart des cas, on est libre d'envisager une architecture n'utilisant pas ces différentes techniques.

Architecture en couches

DDD recommande la structuration du code en couche afin de ne pas diluer la logique métier à plusieurs endroits. Chaque couche à une fonction particulière et est utilisable par d'autres couches de façon à mutualiser le code suivant une logique et éviter ainsi la duplication de code.

La Figure 22 montre les différentes couches et leurs relations, tel que suggéré par l'approche DDD :

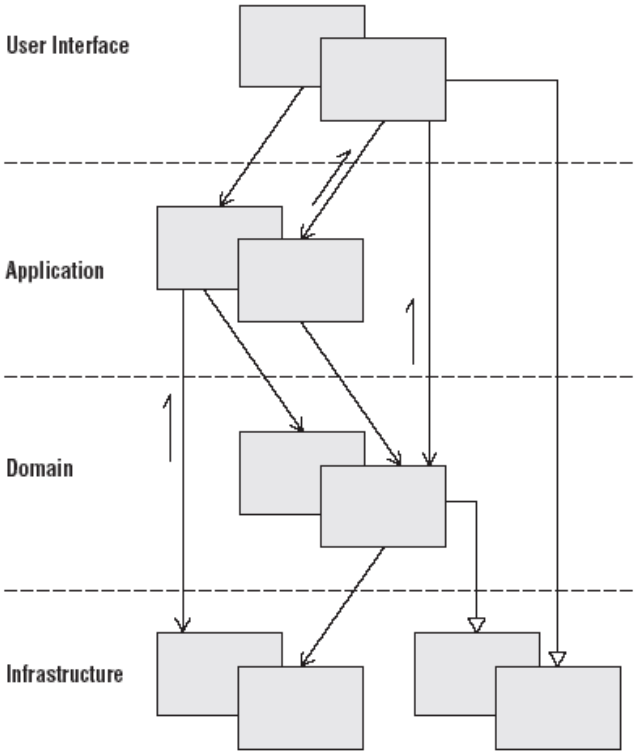

Figure 22 : DDD – Architecture en couches [DDD, 2004]

DDD recommande de développer chaque couche de manière cohésive. Une couche ne peut entretenir des dépendances que vers les couches inférieures (il faut utiliser les patterns standards d'architecture pour assurer un couplage nul vers les couches supérieures).

Le code lié au modèle du domaine doit être concentré dans une seule couche isolée de l'interface utilisateur, de l'application et de l'infrastructure. Les objets du domaine sont alors allégés de leurs responsabilités d'affichage, de persistance, et de gestion des tâches applicatives, et peuvent être

ainsi axés uniquement sur l'expression du modèle métier. Cette approche permet d'isoler la logique métier du reste de l'application. Ceci permet à la couche métier d'être consistante (pas de dilution dans les autres couches), de pouvoir évoluer plus facilement et d'être suffisamment claire pour saisir les connaissances métiers essentielles au logiciel.

Couche utilisateur

La couche utilisateur (user interface) est la couche responsable de montrer l'information à l'utilisateur et d'interpréter les commandes de l'utilisateur (l'utilisateur peut tout aussi bien être un utilisateur humain que logiciel).

- Elle peut effectuer des validations basiques (non liées à des règles métiers) sur les entrées de l'utilisateur avant de les transmettre aux autres couches de l'application.
- Elle ne doit pas contenir de logique métier ni de logique d'accès aux données.
- Cette couche peut faire appel à la couche application et à la couche domaine.

Couche application

La couche application (application layer) sépare la couche utilisateur de la couche domaine.

- Elle permet d'isoler la couche domaine des aspects techniques nécessaires à son fonctionnement.
- Elle peut contenir les services applicatifs et servir d'intermédiaire entre la couche domaine et les objets qui y font appel.
- Elle expose les capacités du système en proposant une abstraction à la logique du domaine contenue dans la couche domaine.
- Elle tend à préserver le domaine en concentrant de nombreux éléments de logique applicative.
- Elle sert d'implémentation concrète à la frontière du contexte borné. Elle peut assurer les échanges avec les autres contextes bornés en utilisant, par exemple, des services REST, des web services ou par l'intermédiaire d'un bus de communication.
- Elle ne contient pas de code métier mais peut être amenée à contenir du code permettant de gérer des changements dans la couche utilisateur.
- Elle ne doit pas garder l'état des objets métier mais peut stocker l'état d'avancement d'une tâche de l'application.
- Elle permet d'effectuer la navigation entre les écrans de l'interface graphique et les interactions avec les couches applications d'autres systèmes.
- Elle peut effectuer des validations basiques (non liées à des règles métiers) sur les entrées de l'utilisateur avant de les transmettre aux autres couches de l'application.
- Elle ne doit pas contenir de logique métier ni de logique d'accès aux données.
- Cette couche peut faire appel à la couche domaine et à la couche infrastructure.

Couche domaine

La couche domaine (domain layer) contient les informations sur le domaine et la logique métier.

- Elle détient tous les concepts du modèle métier, les cas d'utilisation et les règles métiers.
- Elle contient l'état des objets métiers. Toutefois, elle n'effectue pas directement la persistance des objets métiers.
- Elle peut aussi contenir l'état d'un cas d'utilisation métier si celui-ci est formé de plusieurs requêtes de l'utilisateur.
- Elle peut contenir des objets de service si leur comportement ne peut être implémenté dans un objet métier. Les services contiennent des comportements métier qui ne peuvent pas faire partie d'un objet du modèle.
- Cette couche est le cœur du métier, elle doit être isolée des autres couches et ne peut être dépendantes de cadre conceptuel.
- Cette couche ne peut faire appel qu'à la couche infrastructure.

Couche infrastructure

La couche infrastructure (infrastructure layer) fournit les capacités techniques génériques attendues pour le bon fonctionnement des autres couches.

La couche infrastructure masque les détails d'implémentation technique. C'est elle qui va, entre autre, s'occuper de :

- Encapsuler et proposer les capacités de persistance des données du modèle.
- Gérer la communication entre couches.
- Encapsuler et proposer les capacités de message applicatif.
- Etc…

Le bénéfice majeur de l'encapsulation des connaissances d'infrastructure au sein de la couche infrastructure est que la couche domaine ne doit pas du tout être consciente des choix techniques, et de cette façon aucune dépendance n'est introduite avec le modèle.

Entité

DDD définit le concept d'entité en partant du principe que la plupart des objets ne sont pas caractérisés fondamentalement par leurs attributs mais plutôt par leur continuité et leur identité qui s'étend sur la durée de vie d'un système et peut même s'allonger au-delà.

L'identité d'une entité doit être unique et doit le rester durant tous les états du logiciel. Deux entités ne peuvent donc pas partager le même identifiant sous peine de devoir considérer le domaine comme étant dans un état incohérent. Elle peut être construite à partir d'un ou plusieurs attributs de l'entité ou être générée spécifiquement par l'application.

Les entités sont des éléments importants du domaine, elles devraient être repérées dès le début du processus de modélisation. Tous les objets du domaine métier ne seront pas nécessairement des entités. Il est important de déterminer si un objet est une entité ou non, ce dont il est question dans la technique suivante.

Objet-valeur

Le concept d'objet-valeur est complémentaire au concept entité dans le sens où il apporte une solution de design aux objets du domaine qui ne sont pas perçu comme des entités.

Il pourrait être tentant de faire de tous les objets du domaine des entités, mais créer et tracer de l'identité à un coût (coût de gestion et performance). La solution préconisée par DDD est de reconnaître les cas où nous avons seulement besoin de stocker les attributs d'un élément du domaine. L'attention n'est pas portée sur l'identité ni la continuité de l'objet mais bien sur le besoin de connaître la valeur de ses attributs. Un objet qui est utilisé pour décrire certains aspects du domaine et qui n'a pas d'identité est appelé Objet-valeur.

Les objet-valeur ne disposant pas d'une identité propre, ces derniers peuvent facilement être créé et supprimé. Il n'est pas nécessaire de se soucier de leur identité et le garbage collector récupérera l'espace mémoire occupé par l'objet lorsqu'il ne sera plus référencé. Ceci permet de simplifier considérablement le design.

Enfin, DDD signale qu'une bonne pratique est de rendre les objet-valeur immuables. Ils sont créés à l'aide d'un constructeur et jamais modifiés durant leur vie. Si une valeur différente doit être produite pour un objet, il conviendra d'en créer simplement un autre. Les objets immuables sont partageables, ce qui amène beaucoup de retombées en terme de performance. De plus, l'immutabilité permet d'en garantir l'intégrité (car il préserve des risques d'effet de bord en cas de partage et modification de l'objet).

Agrégat

L'agrégat est un pattern de domaine utilisé afin de définir l'appartenance et les frontières des objets.

Un agrégat est un groupe d'objets associés et considéré comme un tout unique vis-à-vis des manipulations de données. L'agrégat est démarqué par une frontière qui sépare les objets se trouvant à l'intérieur de ceux se trouvant à l'extérieur. Chaque agrégat possède une racine. Cette racine est obligatoirement une entité et est le seul objet accessible de l'extérieur. La racine d'un agrégat peut posséder des références vers n'importe quel objet à l'intérieur de ses frontières et les objets à l'intérieur de ses frontières peuvent se référencer entre eux. S'il existe d'autres entités que l'entité racine à l'intérieur de l'agrégat, les identités de ces entités locales à l'agrégat ne peuvent pas être référencées à l'extérieur de ses frontières. Du point de vue des objets extérieurs aux frontières de l'agrégat, seul une référence vers l'objet racine de l'agrégat est autorisée. Et donc les objets à l'intérieur d'un agrégat peuvent contenir des références vers des racines extérieures à l'agrégat auquel ils appartiennent.

Cette isolation des objets d'un agrégat vis-à-vis des objets extérieurs permet de garantir son intégrité car aucun objet extérieur à l'agrégat ne peut modifier directement les objets à l'intérieur de l'agrégat. Seule la racine est modifiable depuis l'extérieur de l'agrégat et est en mesure de modifier les autres objets de l'agrégat. La racine est donc responsable de faire respecter les invariants. Si celle-ci est supprimée alors tous les éléments à l'intérieur de l'agrégat sont supprimés aussi car il n'y aura plus d'autre objet possédant des références vers l'un d'entre eux.

Module

La notion de module permet de regrouper des objets pour en assurer la cohésion. La cohésion recherchée s'applique tant au niveau des relations entre les objets que dans les fonctionnalités gérées par ces objets. La cohésion fonctionnelle est atteinte quand toutes les parties du module travaillent ensemble pour réaliser une tâche bien définie. Elle est considérée comme la meilleure forme de cohésion.

L'utilisation de module permet de proposer une vue d'ensemble du domaine en examinant de manière générale les modules et leurs relations.

Selon DDD, une bonne définition et organisation des modules reflète les caractéristiques suivantes :

- Un module doit former un ensemble de concepts cohérents, de façon à réduire le couplage entre les modules.
- un couplage faible entre module permet de réduire la complexité et d'avoir des modules sur lesquels on peut réfléchir indépendamment.
- Les modules doivent être capables d'évoluer durant la durée de vie du logiciel.
- Les modules doivent être nommés suivant des termes du langage omniprésent.

Service

En principe, lorsqu'on définit le langage omniprésent, les noms des concepts clés permettent de définir les objets du domaine et les fonctionnalités qui sont associées aux noms permettront de définir les comportements de ces objets. Ceux-ci étant le plus souvent directement implémentés dans l'objet.

Cependant, certaines fonctionnalités du domaine ne sont pas naturelles à modéliser sous la forme d'objet. Lorsqu'une fonctionnalité ne peut être associée à un objet (car elle ne peut être qualifiée comme de sa responsabilité) alors un service doit être créé afin de permettre l'implémentation de la fonctionnalité désirée.

La dénomination d'un service et de ses opérations doit mettre l'accent sur la relation qu'il entretient avec d'autres objets. Contrairement aux entités et aux objet-valeur, il est défini uniquement dans des termes exprimant ce que le service peut faire. Dès lors, un service a tendance à être nommé par/pour une activité, plutôt qu'une entité : un verbe plutôt qu'un nom.

La granularité du service entre un service domaine ou applicatif est parfois difficile à discerner. Les services des couches domaine et application sont généralement tous deux construits autour des entités et des objets-valeur du domaine. Ceux-ci fournissent des fonctionnalités directement reliées à ces objets. Si l'opération effectuée appartient conceptuellement à la couche application alors c'est dans la couche application que le service devrait être placé. Si l'opération concerne des objets du domaine, si elle est strictement liée aux objets d'un domaine particulier et répond à un besoin du domaine alors elle devrait appartenir à la couche domaine.

Selon DDD, un service (application ou domaine) présente trois caractéristiques :

- L'opération dans le service fait référence à un concept du *domaine* qui n'appartient pas naturellement à une entité ou à un objet-valeur.
- L'interface du service est définie en utilisant des termes et des objets issus du domaine.
- Le service n'a pas d'état (stateless).

Exemple : Partitionnement des services au travers des couches domaine/application/infrastructure.

Cet exemple issu de [DDD, 2004] concerne la répartition des responsabilités et des services dans le cadre d'une application de transfert de fonds entre deux comptes client.

Application **Service applicatif de transfert de fonds**	• Traite les entrées (ex : une requête XML) • Envoi des messages au service domaine pour exécution • Ecoute la confirmation des exécutions demandées • Décide d'envoyer une notification à l'aide du service d'infrastructure
Domaine **Service domaine de transfert de fonds**	• Interagit avec les objets domaine nécessaires (compte, …) et effectue les débits et les crédits appropriés • Fournit une confirmation du résultat (transfert autorisé ou non, etc.)
Infrastructure **Service d'envoi de notification**	• Envoi de mails, courriers ou tout autre moyen de communication tel que requis par l'application

Fabrique

La fabrique est un pattern de domaine qui permet d'encapsuler le processus de création d'objets, et est particulièrement utile pour la création des agrégats.

Elle permet de centraliser et de masquer la connaissance nécessaire à la construction de structure complexe et ainsi éviter que la logique de création des objets ne se trouve dans l'agrégat. De plus, la gestion des identités des entités n'est pas forcément triviale. Des objets peuvent être créés à partir de rien ou ils peuvent aussi avoir déjà existé, et il peut être nécessaire d'effectuer des traitements pour récupérer les données de l'entité par exemple à partir d'une base de données.

Il est recommandé de déléguer à une fabrique la création d'un agrégat de façon atomique.

Néanmoins, l'utilisation de fabriques n'est pas indispensable, DDD recommande de privilégier un constructeur simple quand :

- La construction n'est pas compliquée : pas d'invariants, de contraintes, de relations avec d'autres objets.
- La création n'implique pas la création d'autres objets et que toutes les données membres sont passées par le constructeur.
- Il n'y a pas de nécessité de choisir parmi plusieurs implémentations concrètes.

Entrepôt

L'entrepôt est un pattern du domaine qui permet d'encapsuler la logique nécessaire à l'obtention de références d'objet. De cette façon, les objets du domaine n'ont pas à se soucier de la récupération des références aux autres objets du domaine dont ils ont besoin. Les entrepôts contiennent (et masquent) les informations détaillées permettant d'accéder à l'infrastructure de stockage. Ceux-ci proposent aux objets du domaine une interface simple qui leur permet de récupérer des références à des objets du domaine sur base de divers critères de sélection.

Il existe une relation particulière entre les fabriques et les entrepôts dans le sens où ces deux patterns du domaine participent au cycle de vie des objets du domaine. Il ne faut pas néanmoins confondre les deux patterns. Au niveau de la création des objets, la fabrique crée des objets du domaine à partir de rien, tandis que l'entrepôt crée des objets en les récupérant dans l'endroit de stockage adéquat. L'objet restitué ayant déjà existé, son identité ne doit pas être créée.

Evénement du domaine

Evénement du domaine est un pattern qui permet de représenter un changement d'état d'éléments du domaine et considéré comme important par les experts.

Ce pattern tente d'apporter une solution aux deux problématiques suivantes :

- Les entités n'ont aucune connaissance relative à la raison de leur changement d'état. En effet, une entité est responsable du suivi de son état et des règles qui régissent son cycle de vie, mais ceci ne permet pas de connaître les causes réelles de ses changements. Ce manque d'information empêche la compréhension des éléments ayants conduit un système dans un état particulier.

- L'état d'un système distribué ne peut pas être totalement cohérent en tout temps. Bien que les agrégats soient constamment gardés cohérents en interne, d'autres modifications sont réalisées dans un système distribué, ceci de manière asynchrone. Au fur et à mesure, les changements se propagent à travers des nœuds du réseau, et il peut s'avérer compliqué de définir l'état global du système distribué ou de résoudre plusieurs mises à jour arrivant dans un ordre aléatoire ou depuis des sources distinctes.

L'approche DDD recommande de modéliser les informations relatives aux activités du domaine en tant que série d'événements. Chacun d'eux est représenté en tant qu'objet du domaine. Il devient ainsi une partie intégrante du modèle, une représentation de quelque chose qui a eu lieu dans le domaine. Il devient alors possible de connaître les éléments ayant conduit le système dans un état particulier. Dans le cas d'un système distribué, l'état d'une entité peut être déduit des événements du domaine actuellement connus par un nœud particulier, ce qui permet de restituer un modèle global cohérent.

Les événements du domaine sont ordinairement immuables, étant donné qu'ils sont le récit de quelque chose qui s'est déroulé dans le passé.

Un événement de domaine contient généralement les informations suivantes :

- description
- moment de survenance
- identité des entités impliquées
- moment de l'entrée de l'événement dans le système
- identité de l'acteur à l'initiative

Notons que les évènements du domaine sont distincts des événements du système qui reflètent eux l'activité dans le logiciel lui-même. Bien que souvent un événement système soit associé à un événement du domaine, ceci dans le cadre d'une réponse à l'événement de domaine, ou comme moyen de transporter des informations sur l'événement du domaine dans le système.

Partie 2 : Intégration du style d'architecture DDD au sein de TOGAF

L'objectif suivi est de mettre en regard TOGAF et DDD afin de mieux comprendre comment TOGAF permet de soutenir la mise en œuvre d'une approche DDD, et de voir s'il y a lieu d'adapter TOGAF afin d'englober une telle approche.

Actuellement, TOGAF ne propose pas de support pour l'intégration du style d'architecture DDD et aucune documentation ou article scientifique permettant d'aller dans ce sens n'a encore vu le jour. Nous allons donc vérifier que TOGAF soit bien compatible avec cette approche.

A priori, l'adressage de TOGAF à un style d'architecture particulier ne devrait pas nécessiter d'adaptation importante. En effet, TOGAF préconise durant sa phase préliminaire de procéder à une définition de la capacité d'architecture désirée. Cette capacité d'adaptation de TOGAF lui permet de répondre à des exigences d'architecture spécifiques. La capacité d'adaptation proposée traduit le caractère englobant de TOGAF par rapport aux styles d'architecture en général.

Compte tenu des éléments abordés précédemment, la démarche de recherche choisie est d'exécuter l'étape d'adaptation de la Capacité d'Architecture, tel que recommandée par TOGAF. De cette façon, nous pourrons analyser la faculté de TOGAF à soutenir l'intégration du style d'architecture DDD, et lorsque cela sera jugé nécessaire, les adaptations requises de TOGAF seront exprimées.

2.1 Exécution de la phase préliminaire – Définition de la Capacité d'Architecture

Cette phase est la première réalisée dans le cadre d'une démarche d'architecture menée à l'aide de TOGAF. C'est durant cette phase que le contenu et la forme de l'architecture est précisée.

La démarche d'adoption du style d'architecture DDD par TOGAF prend véritablement racine dans la phase préliminaire qui est en grande partie concentrée sur l'adaptation du cadre d'architecture. Les éléments nécessaires à l'adoption du style DDD seront a priori définis à la sortie de cette phase.

La Figure 23 présente les étapes réalisées durant la phase préliminaire ainsi que les livrables généralement attendus à la sortie de ces étapes. Pour chacune d'elles, une évaluation de l'impact de l'intégration de DDD est réalisée.

Etapes	Livrables	Niveau d'adaptation nécessaire*
(Re) Formaliser les besoins d'architecture d'entreprise	Les Principes du business, buts du business et moteurs du business	0
Identifier et établir les principes d'architecture	Catalogue des Principes d'Architecture	1
Définir le périmètre des organisations d'entreprise concernées.	Le Modèle d'Organisation pour l'Architecture d'Entreprise	0
Vérifier les cadres de gouvernance et de support.	Le Modèle d'Organisation pour l'Architecture d'Entreprise	0
Définir, établir l'équipe et l'organisation de l'architecture d'entreprise.	Le Modèle d'Organisation pour l'Architecture d'Entreprise	0
Sélectionner et adapter un ou des cadre(s) d'architecture	Le Cadre d'Architecture Contextualisé	2
Alimenter le référentiel d'architecture	Le Référentiel d'Architecture	1
Mettre en œuvre des outils d'architecture	Les Outils d'Architecture	0
Initier les travaux d'architecture	La Demande de Mise en Chantier d'Architecture	0

Figure 23 : Niveau d'impact de l'intégration de DDD sur les étapes et la phase préliminaire de TOGAF

* Niveaux de classification de l'adaptation nécessaire à l'intégration du style d'architecture DDD :

- **0** : pas de prise en compte d'élément particulier à DDD
- **1** : prise en compte d'élément particulier à DDD sans impact sur le cadre d'architecture
- **2** : prise en compte d'élément particulier à DDD avec impact sur le cadre d'architecture

Les sections suivantes présentent les détails relatifs aux impacts de l'intégration de DDD identifiés dans la Figure 23.

2.1.1 Etape : Identifier et établir les principes d'architecture

L'exécution de la phase préliminaire requiert de définir les principes d'architecture qui feront partie des contraintes sur tout travail d'architecture réalisé dans l'entreprise. Les principes d'architecture définissent les règles générales et les lignes directrices concernant l'utilisation et le déploiement de toutes ressources et actifs informatiques au sein de l'entreprise. Les principes reflètent un niveau de consensus parmi les différents éléments de l'entreprise et constituent la base permettant la prise de décision d'architecture future. Ils sont également utilisés pour évaluer et valider les résultats de l'architecture, les décisions prises et constituent également une aide à la gouvernance.

Le livrable issu de cette étape est le catalogue des Principes d'Architecture. Celui-ci capture les principes du métier et de l'architecture qui décrivent l'apparence d'une « bonne » solution ou architecture.

La définition et l'adoption d'un principe consacré à l'application du style d'architecture DDD est le point de départ permettant d'introduire ce style d'architecture au sein de TOGAF.

La définition du principe d'application de DDD doit suivre les recommandations de TOGAF en matière de contenu et de qualité des principes. Pour supporter cette démarche de définition de principe, TOGAF propose, dans sa partie PARTIE III : « Recommandations et Techniques ADM : Principes d'Architecture », un ensemble de règles et de bonnes pratiques soutenant la définition des principes d'architecture.

En nous référant aux recommandations TOGAF et aux caractéristiques du style d'architecture DDD les principes suivants peuvent être définis :

Nom	Appliquer le Style d'Architecture DDD.
Déclaration	L'application du style d'architecture DDD permet à l'entreprise et aux démarches de conception d'architecture qui y sont réalisées de porter une grande attention au domaine métier de l'entreprise.
Justification	L'application du style d'architecture DDD permet d'accorder une grande importance au domaine métier de l'entreprise et ainsi de valoriser ce qui est considéré comme la réelle plus-value de l'entreprise et de son système d'information.

Implications	• L'application de ce principe requiert l'application des principes suivants :
	  ○ Appliquer les techniques de compréhension et de communication du domaine tel que recommandées par le style d'Architecture DDD.
	  ○ Appliquer les techniques de préservation de l'intégrité du modèle domaine tel que recommandées par le style d'Architecture DDD.
	  ○ Adopter une conception dirigée par le modèle domaine tel que recommandée par le style d'Architecture DDD.
	• Les ressources nécessaires à l'application de ce principe doivent être identifiées et disponibles.

La considération complète et précise de ce principe conduit à l'application de principes supplémentaires consacrés aux différentes techniques requises. Ceci permettra d'en exprimer les caractéristiques et de lever toute ambiguïté concernant leur nature et l'objet de leur application.

Nom	Appliquer les techniques de compréhension et de communication du domaine recommandées par le style d'architecture DDD.
Déclaration	L'application des techniques de compréhension et de communication du domaine recommandées par le style d'architecture DDD permet à tous les acteurs impliqués dans la démarche de conception logicielle de comprendre en profondeur le domaine métier.
Justification	La compréhension en profondeur du domaine métier par tous les acteurs impliqués dans la conception logicielle est nécessaire afin que le logiciel devienne le reflet du métier et lui apporte une réelle plus-value.
Implications	• L'adoption de ce principe sous-entend l'adoption des principes suivants : ○ Appliquer le concept de « Langage omniprésent ». ○ Appliquer les techniques et recommandations de distillation du domaine métier.

Nom	Appliquer les techniques de préservation de l'intégrité du modèle domaine recommandées par le style d'architecture DDD.
Déclaration	L'application des techniques de préservation de l'intégrité du modèle recommandées par le style d'architecture DDD permet d'organiser la segmentation du domaine métier de l'entreprise en différents modèles et de garantir leur intégrité.
Justification	La segmentation du domaine métier en plusieurs domaines permet de faciliter l'organisation du travail collaboratif et d'adapter la portée des modèles en regard des spécificités organisationnelles de l'entreprise ou de la nature des données. Les techniques de préservation de l'intégrité des modèles permettent de définir et d'organiser les relations entre ces modèles, ceci afin de fournir une vision d'ensemble et de faciliter l'intégration des différents modèles entre eux.
Implications	• L'application de ce principe sous-entend l'application des principes suivants : ○ Appliquer le concept de contexte borné. ○ Appliquer le principe d'intégration continue. ○ Appliquer les stratégies de préservation de l'intégrité des modèles.

Nom	Appliquer une approche de conception dirigée par le modèle telle que recommandée par le style d'architecture DDD.
Déclaration	L'application d'une approche de conception dirigée par le modèle telle que recommandée par le style d'architecture DDD permet de concentrer les efforts de production sur le résultat de la modélisation.
Justification	L'application d'une approche de conception dirigée par le modèle permet de lier étroitement la modélisation du domaine et la conception logicielle. La conception dirigée par le modèle recommande de concevoir le logiciel de telle sorte qu'il reflète le modèle du domaine de manière très littérale, de cette façon le code devient l'expression du modèle.
Implications	• Cette approche nécessite une implication des développeurs très tôt dans le processus de modélisation. Ceci permet d'assurer que tout le monde comprenne les subtilités du modèle et de vérifier l'implémentatibilité du modèle en code. De cette façon, le modèle peut, si besoin, être adapté pour faciliter son implémentation en tenant compte des limites introduites par les outils de développement. • Cette approche nécessite également une implication de l'analyste dans le processus de conception logicielle, ceci afin de lui permettre d'être informé des préoccupations liées aux limites introduites par le développement et de recueillir les refactorisations du code (via émergence ou évolution des exigences) qui conduiront la plupart du temps à un changement du modèle. Si tel n'est pas le cas, il existe un risque que le modèle logiciel cesse d'exprimer le modèle original et perde, dès lors, sa valeur fondamentale. • L'application de ce principe sous-entend l'application des principes suivants :

	○ Appliquer l'architectures en couche telle que recommandée par DDD. ○ Appliquer le concept d'entité. ○ Appliquer le concept d'objet-valeur. ○ Appliquer le concept d'agrégat. ○ Appliquer le concept de module. ○ Appliquer le concept de service. ○ Appliquer le concept de fabrique. ○ Appliquer le concept d'entrepôt. ○ Appliquer le concept d'évènement du domaine.

En complément de ces quelques principes de haut niveau relatifs à l'application de DDD, d'autres principes plus fins devraient être produits afin de préciser d'avantage l'usage des techniques et recommandations issues de DDD au sein de la démarche d'architecture d'entreprise.

Nom	Appliquer le concept de langage omniprésent tel que recommandé par le style d'architecture DDD.
Déclaration	…
Justification	
Implications	

…

Les principes d'architecture présentés dans cette section reflètent une volonté d'intégration complète de DDD au sein de la Capacité d'Architecture. Cependant, selon les cas, il se peut qu'en fonction du degré d'intégration désiré, certaines capacités d'architecture n'adoptent pas toujours l'entièreté de ces principes mais seulement certains d'entre eux. Il conviendra alors de sélectionner et, si besoin, d'adapter les principes DDD précédemment décrits. Il faudra tenir compte des impacts de la non-application de certaines recommandations et techniques afin d'évaluer une diminution possible des résultats attendus.

L'hypothèse de la non-application d'une partie des principes d'architecture relatifs à DDD a été exclue dans le cadre de ce mémoire. La documentation de cette forme d'intégration devra se faire au cas par cas en fonction des principes sélectionnés.

Pour conclure, cette étape ne nécessite pas d'adaptation particulière pour permettre l'intégration du style d'architecture DDD. L'intégration de DDD dans cette étape se concrétise au travers de la définition de principes d'architecture spécifiques à DDD, formalisant sa mise en œuvre au sein de la démarche d'architecture d'entreprise.

2.1.2 Etape : Sélectionner et adapter un ou des cadre(s) d'architecture

TOGAF suggère dans son document une série de cadres permettant de soutenir la documentation des différents aspects de l'architecture d'entreprise. L'examen de ces cadres est nécessaire afin de vérifier qu'ils soient adaptés à la Capacité d'Architecture désirée.

Les différents cadres TOGAF qui feront l'objet d'un examen sont :

- Le cadre de Contenu
  - Le cadre de Concepts
  - Le cadre de Vues

- Le cadre de Capacité

Inspection du cadre de contenu

TOGAF fournit un cadre de contenu (cadre de concepts et cadre de vues).

- Le **cadre de concepts** (métamodèle TOGAF) a pour objectif d'apporter une structure et une définition formelle des éléments de base disponibles ainsi que de leurs relations.

- Le **cadre de vues** a pour objectif d'apporter une structure et une définition des vues et des modèles entrants/sortants en regard des différentes phases du cycle ADM.

Une des tâches de l'architecte est d'adapter le cadre de contenu en tenant compte des spécificités de l'entreprise et de la démarche d'architecture, ceci y compris en tenant compte des principes d'architecture définis.

L'intégration des principes DDD au sein des principes d'architecture nécessite une adaptation du cadre de contenu. Ceci inclut :

- Une adaptation du cadre de concept, afin de documenter les éléments de base spécifiques à l'extension DDD. Ceux-ci feront l'objet d'une incorporation au sein du cadre de concept TOGAF. Pour cela, une définition de ces éléments et des relations qu'ils entretiennent avec les éléments de base de TOGAF sera effectuée.

- Une adaptation du cadre de vues, afin de documenter les nouvelles vues ou les modifications de vues existantes. Ces changements doivent tenir compte des nouveaux éléments de base spécifiques à DDD et des relations introduites suite à l'intégration de ces éléments dans le métamodèle TOGAF.

Le travail d'adaptation du cadre de contenu de TOGAF à l'approche DDD peut se décomposer en plusieurs étapes :

- Identifier la forme revêtue par l'adaptation permettant de soutenir l'approche DDD.

- Documenter l'adaptation permettant de soutenir l'approche DDD.

- Identifier et formaliser les éléments de base DDD nécessaires à la documentation des architectures désirant appliquer les principes DDD.

- Rapprocher les éléments de base TOGAF et DDD similaires et identifier les nouvelles relations entretenues entre ces éléments.

- Etendre le métamodèle TOGAF avec les éléments de base nécessaires à l'adoption de la démarche DDD.

- Rapprocher les éléments d'extension TOGAF et DDD similaires et identifier les nouvelles relations entretenues entre ces éléments.

- Etendre le métamodèle TOGAF accompagné de ses extensions avec les éléments de base nécessaires à l'adoption de la démarche DDD.

- Identifier les nouveaux artefacts d'architecture en regard des phases de l'ADM, ceci en proposant les modèles et les vues nécessaires permettant de satisfaire les parties prenantes vis-à-vis des préoccupations spécifiques à la démarche DDD.

Forme de l'adaptation

Pour rappel, le métamodèle de base TOGAF est un modèle présentant le jeu de caractéristiques minimal attendu par toute démarche d'architecture d'entreprise.

Ce modèle est capable de supporter l'inclusion d'extensions optionnelles issues des décisions d'architecture effectuées durant la confection de l'engagement d'architecture. TOGAF propose dans son document six extensions. Celles-ci permettent la prise en charge d'une modélisation plus spécifique ou plus approfondie de domaines d'intérêt particuliers. Une extension regroupe de manière logique un ensemble d'éléments de base, des modèles et des points de vue.

Les extensions du métamodèle de base de TOGAF sont illustrées dans la Figure 24 :

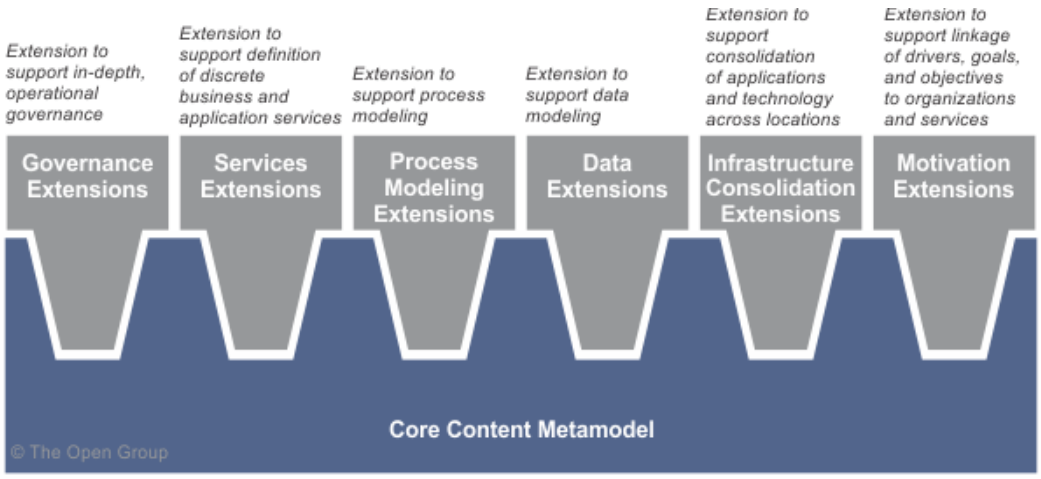

Figure 24 : TOGAF – Extensions du métamodèle TOGAF [TOGAF9.1]

Les extensions proposées ne sont qu'une suggestion et une adaptation supplémentaire du métamodèle peut être effectuée, ceci en fonction des besoins spécifiques de la démarche d'AE.

La capacité d'évolution du métamodèle de base de TOGAF est une caractéristique pertinente dans le cadre de ce mémoire visant la documentation de l'intégration du style d'architecture DDD au sein de TOGAF. En effet, afin de supporter l'intégration désirée, il semble raisonnable de proposer une extension du métamodèle TOGAF spécifique à DDD. Pour être utilisable au sein de TOGAF, l'extension visée abordera de façon précise les éléments suivants :

- Définition générale de l'extension DDD (situations d'application, bénéfices, localisation…).
- Définition des éléments de base spécifiques à l'extension DDD.
- Intégration des éléments de base DDD au sein du métamodèle TOGAF.
- Intégration des éléments de base DDD vis-à-vis des extensions du métamodèle TOGAF.

Définition générale de l'extension DDD

L'extension de modélisation « Domain Driven Design Extensions » est destinée à permettre l'adoption du style d'architecture de conception dirigée par le domaine au sein du cadre de contenu de TOGAF (voir Figure 25).

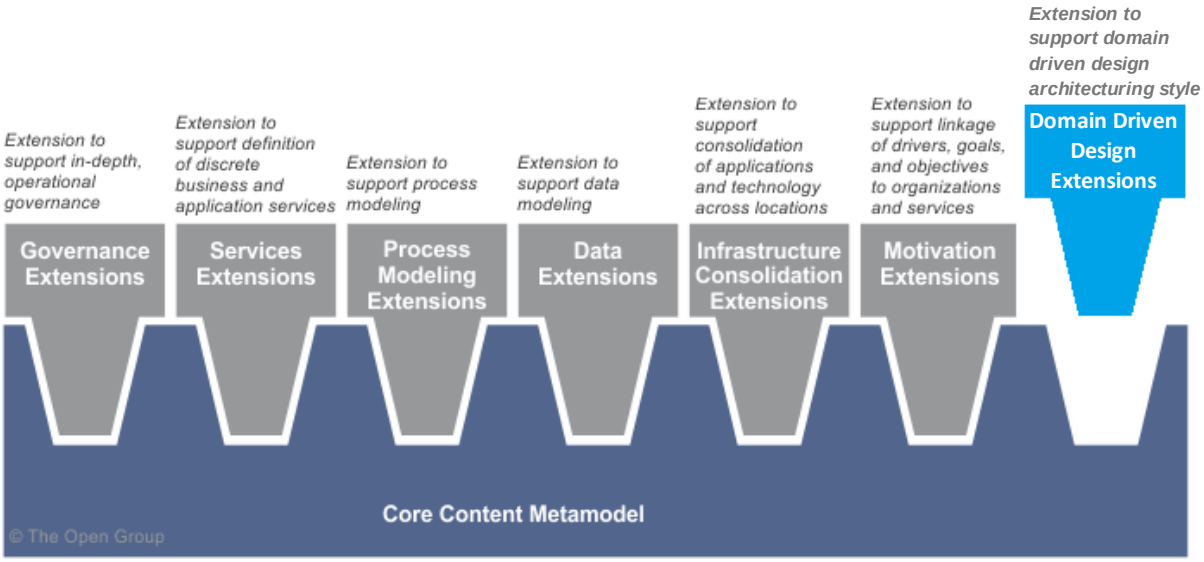

Figure 25 : TOGAF – Domain Driven Design Extensions

Description

L'extension DDD : « Domain Driven Design Extensions », regroupe et structure les éléments nécessaires à l'adoption du style d'architecture de conception dirigée par le domaine au sein de TOGAF. Elle ajoute des éléments de base spécifiques au métamodèle de TOGAF et fournit un ensemble de nouveaux artefacts d'architecture classifiés en fonction des différentes phases ADM concernées par leur réalisation.

Localisation

La Figure 26 présente la localisation de l'extension et sa portée au sein des subdivisions du métamodèle TOGAF. De par sa nature et ses considérations spécifiques, l'adoption de l'extension DDD au sein de TOGAF influencera principalement le domaine d'architecture des systèmes d'informations, (architecture des données et l'architecture applicative : phase C du cycle ADM).

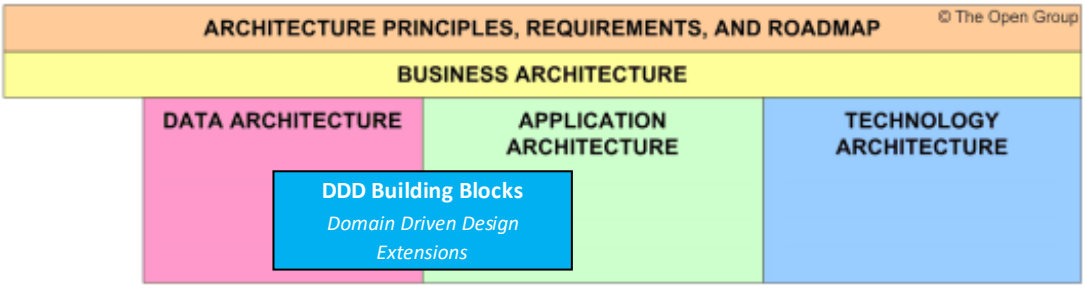

Figure 26 : TOGAF – Localisation de l'extension DDD dans le métamodèle TOGAF

Portée

- Création de contextes bornés qui encapsulent un ensemble d'agrégat regroupés en module, ceci en tenant compte de leurs natures et de leurs dépendances.
- Création d'agrégats afin de regrouper des entités de données et considérer celles-ci comme un tout unique vis-à-vis des manipulations de données.
- Identification des racines d'agrégats.
- Spécialisation des objets de données sous la forme d'entité ou d'objet-valeur.
- Création de fabriques qui encapsulent les connaissances nécessaires à la création des agrégats.
- Création de dépôts responsables de la persistance et de la récupération des données dans l'endroit de persistance adéquat.
- Création de services applicatifs et de domaine encapsulant les fonctionnalités métier requises.
- Création de divers modèles se concentrant sur : l'organisation des contextes bornés, l'organisation des agrégats, la classification et les dépendances des entités de données.

Situation d'utilisation

- Aborder de façon pragmatique l'étude de domaine métier complexe et/ou de taille importante.
- Aider à la fragmentation de l'entreprise afin de faciliter le travail parallèle et collaboratif des équipes de développement logiciel.
- Aider à produire des solutions d'architecture désirant un couplage faible et maitrisé entre les différents segments de données de l'entreprise.
- Volonté de l'entreprise de capitaliser ses données métier.
- Les regroupements logiques de données peuvent être utilisés pour définir des limites de gouvernance, de sécurité ou de déploiement, ce qui permet une appréciation beaucoup plus holistique des problèmes de données entourant l'architecture.

Formalisation des éléments de base de l'extension DDD

L'extension DDD contient les éléments de base supportant la description de l'architecture dans le cadre d'une approche de conception dirigée par le domaine. Les éléments sont formalisés sous la forme d'un diagramme de classe exposant ces derniers et leurs relations. En plus de cela, une définition de la sémantique de chaque élément est fournie afin de permettre une compréhension commune et sans aucune ambiguïté de ces éléments.

Le diagramme de classe exprimé dans la Figure 27 présente les éléments de base de l'extension DDD et leurs relations :

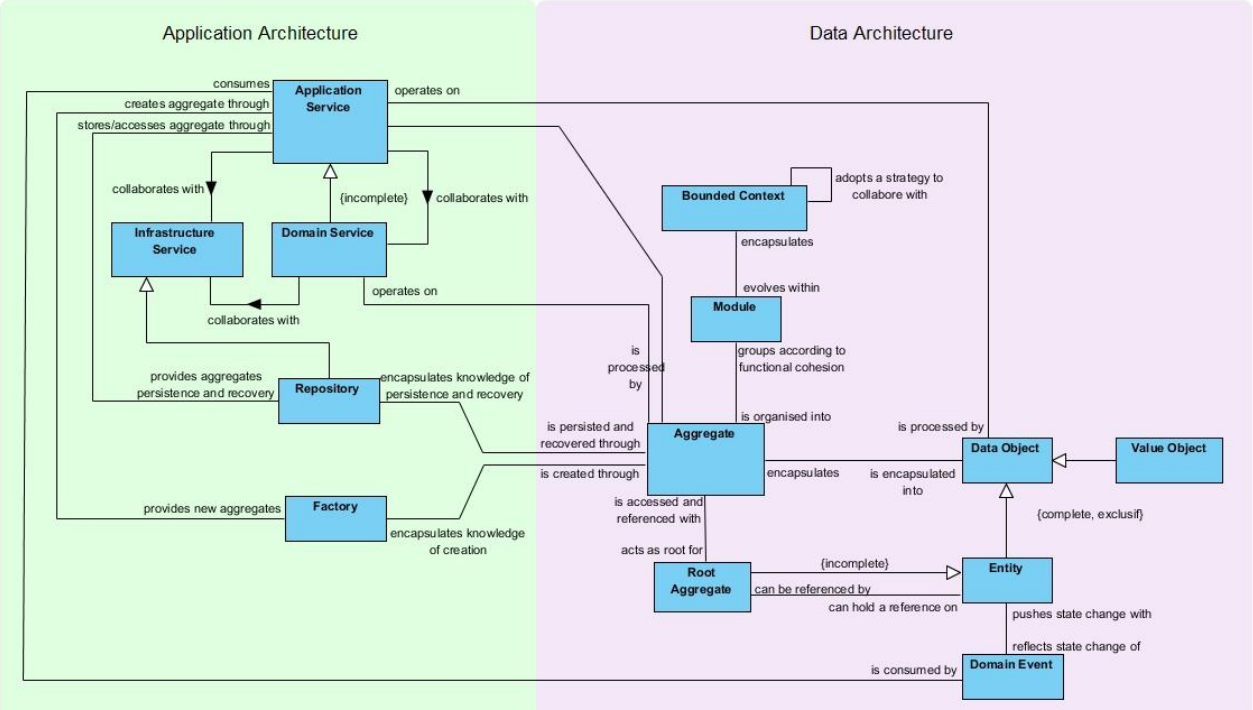

Figure 27 : DDD – Métamodèle des éléments de base de l'extension DDD

Les éléments de base et les relations présentés dans la Figure 27 sont issus des « techniques de préservation de l'intégrité du modèle » et des « techniques de conception dirigée par le domaine ».

Les éléments de base de l'extension DDD peuvent être regroupés en deux catégories distinctes :

- Les éléments de base relatifs à l'architecture des données. C'est-à-dire les éléments concernés par l'aspect statique du système d'information. L'intérêt de ces éléments de base est porté sur la classification et la structure des données du système d'information.

- Les éléments de base relatifs à l'architecture applicative. C'est-à-dire les éléments concernés par l'aspect dynamique du système d'information. L'intérêt de ces éléments de base est porté sur la classification et l'organisation des fonctionnalités applicatives du système d'information.

En complément de la Figure 27, le tableau suivant décrit les différents éléments de base de l'extension DDD. Ces descriptions documentent les caractéristiques et les relations de ces différents éléments de base les uns vis-à-vis des autres.

Dénomination	Description
Aggregate	Un **agrégat** est un ensemble d'objets de données (voir Data Object) regroupés de façon logique en regard de leurs relations et de leur nature. Cette association est considérée comme un tout unique vis-à-vis des modifications de données.
is accessed and referenced by	Chaque agrégat possède une entité racine (voir Root Aggregate).
is processed by	Un agrégat est manipulé par un service domaine (voir Domain Service).
is created through	Un agrégat est créé à l'aide d'une fabrique (voir Factory).
is persisted and recovered through	Un agrégat est persisté et récupéré à l'aide d'un dépôt (voir Repository).
is organised into	Un ensemble d'agrégats, qui partagent une cohésion fonctionnelle, peuvent être regroupés au sein d'un module (voir Module).
Application Service	**U**n **service applicatif** est une encapsulation d'une fonctionnalité applicative faisant référence à un concept du domaine qui n'appartient pas naturellement à un agrégat (voir Aggregate).
operates on	L'accomplissement de la fonctionnalité nécessite généralement une manipulation d'objets de données issus de plusieurs agrégats.
collaborates with collaborates with creates aggregate through stores/accesses aggregate through	Pour cela le service applicatif interagit avec : • Des services du domaine (voir Domain Service), afin de manipuler des agrégats et obtenir des objets de données. • Des services d'infrastructure (voir Infrastructure Service), afin d'utiliser des éléments d'infrastructure particuliers (ex : envoi Email). • Des fabriques (voir Factory) pour créer de nouveaux agrégats. • Des dépôts (voir Repository) afin de récupérer des agrégats ayant déjà existés et de faire persister de nouveaux agrégats ou de mettre à jour des agrégats.
consumes	Un service applicatif est capable de traiter des événements du domaine (voir Domain Event) afin d'appliquer les activités de compensation nécessaires en fonction des attributs de l'événement traité.
	Un service applicatif peut être spécialisé en service du domaine (voir Domain Service).
	Un service applicatif ne possède pas d'état.

Dénomination	Description
Bounded Context	Un **contexte borné** représente un cadre logique au sein duquel un modèle particulier est défini et applicable. La frontière d'un contexte borné définit la portée d'un modèle mais aussi les limites en termes d'organisation d'équipe, d'utilisation du modèle et de ses manifestations physiques telles que le code source et les schémas de base de données.
encapsulates	Un contexte borné englobe des modules (voir Module).
adopts a strategy to access	Deux contextes bornés peuvent entretenir une relation de dépendance. Dans ce cas, le choix d'une stratégie est nécessaire afin d'organiser la relation de ces deux contextes bornés.
	Un contexte borné emploie un et un seul langage omniprésent (qui peut être partagé entre plusieurs contextes bornés).
Data Object	Un **objet de données** est un ensemble de données reconnues comme une chose par un expert du domaine.
	Un objet de donnée peut être spécialisé en Entité (voir Entity) ou en objet-valeur (voir Value Object).
encapsulates	Un objet de donnée est encapsulé au sein d'un agrégat (voir Aggregate).
Domain Service	Un **service du domaine** est le résultat de la spécialisation d'un service applicatif (voir Application Service). Cette spécialisation se justifie lorsque la portée fonctionnelle de l'opération se limite au cadre d'un ou plusieurs agrégats de même type. Un service du domaine représente l'encapsulation d'une fonctionnalité du domaine et faisant référence à un concept qui n'appartient pas naturellement à un agrégat particulier (voir Aggregate).
collaborates with	Un service du domaine peut collaborer avec des services d'infrastructure (voir Infrastructure Service) afin d'utiliser des éléments d'infrastructure particuliers (ex : envoi Email).
consumes	Un service du domaine est capable d'interpréter des événements du domaine (voir Domain Event) afin d'appliquer les activités de compensation nécessaires, ceci en fonction de la nature de l'événement traité.
	Un service du domaine ne possède pas d'état.

Dénomination	Description
Domain Event reflects state change of is consumed by	Un **évènement du domaine** est une représentation d'un changement d'état d'une ou plusieurs entités (voir Entity) et qui relève potentiellement un intérêt particulier de la part d'autres éléments du domaine. Les informations minimum nécessaires relatives au changement d'état sont : • Heure de survenance • Identités des entités concernées • Nature / Lieu Un évènement du domaine communique toute l'information nécessaire afin d'en permettre une interprétation correcte du changement d'état concerné. Un évènement du domaine est consommé par les services applicatifs (voir Application Service) intéressés. Un évènement du domaine est immutable (car il représente un élément du passé).
Entity pushes state change with	Une **entité** est une spécialisation d'un objet de donnée (voir Data Object) qui n'est pas définie par ses attributs mais plutôt par une continuité et une identité qui s'étendent dans le temps. Une entité possède une identité unique et non-ambigüe. Une entité possède un état. Un changement d'état d'une entité peut être communiqué aux autres éléments du domaine, ceci en faisant l'objet d'un évènement du domaine (voir Domain Event). Une entité peut être spécialisée en racine d'agrégat (voir Root Aggregate).
Factory provides new aggregates encapsulates knowledge of creation	Une **fabrique** est une encapsulation logique des connaissances nécessaires à la création d'un agrégat (voir Aggregate) particulier. Une fabrique est utilisée par les services applicatifs (voir Application Service) pour créer de nouveaux agrégats. Une fabrique encapsule les connaissances nécessaires à la création de nouveaux agrégats.
Infrastructure Service collaborates with	Un **service d'infrastructure** représente l'encapsulation d'une fonctionnalité de soutien. Par exemple : implémenter la persistance des objets métier, exposer des moyens de communication, etc. Un service d'infrastructure ne possède pas d'état. Un service d'infrastructure peut être invoqué par les services applicatifs (voir Application Service) et les services du domaine (voir Domain Service).

Dénomination	Description
Module	Un **module** est utilisé pour organiser les éléments d'un modèle. Cette méthode d'organisation permet de regrouper des agrégats (voir Aggregate) en fonction de leur cohésion fonctionnelle, ceci en vue de réduire la complexité du modèle et ainsi en faciliter la compréhension et la gestion.
groups according to functional cohesion	Un module regroupe un ensemble d'agrégats (voir Aggregate) partageant une certaine cohésion fonctionnelle.
evolves within	Un module évolue au sein d'un contexte borné (voir Bounded Context).
Repository	Un **dépôt** est une encapsulation logique des connaissances d'infrastructure nécessaires à la persistance et à la récupération d'agrégats (voir Aggregate).
	Un dépôt est le résultat de la spécialisation d'un service d'infrastructure (voir Infrastructure Service).
	Un dépôt ne possède pas d'état.
provides aggregates persistence and recovery	Un dépôt est utilisé par les services applicatifs (voir Application Service) pour récupérer ou faire persister des agrégats.
encapsulates knowledge of persistence and recovery	Un dépôt encapsule les connaissances d'infrastructure nécessaires à la persistance et à la récupération d'agrégats (voir Aggregate).
Root Aggregate	Une **racine d'agrégat** est une entité (voir Entity) particulière identifiée au sein d'un agrégat (voir Aggregate). Elle est la seule entité accessible et référençable à l'extérieur de l'agrégat. Les entités à l'intérieure d'un agrégat sont autorisées à détenir des références vers les racines d'autres agrégats.
	Une racine d'agrégat est le résultat de la spécialisation d'une entité.
acts as root for	Une racine d'agrégat agit en tant que racine pour un agrégat.
can be referenced by	Une racine d'agrégat peut être référencée par des entités en dehors de l'agrégat dont elle est la racine.
Value Object	Un **objet-valeur** est une spécialisation d'un objet de donnée (voir Data Object), qui est utilisé pour décrire certains aspects du domaine qui ne nécessitent pas de disposer d'une identité. Ce qui importe avec ce type d'objet, ce sont ses attributs et leurs valeurs.
	Un objet-valeur est immutable, il est créé et plus jamais modifié au cours de sa vie. Il peut donc être facilement partagé.
	Un objet-valeur est le résultat de la spécialisation d'un objet de données (voir Data Object).

Identification des similarités et des relations entre éléments de base DDD et TOGAF

Cette section se concentre premièrement sur la définition de la méthode de comparaison utilisée pour identifier les similarités et relations pouvant être établies entre les éléments de base issus de TOGAF et ceux provenant de DDD. Deuxièmement, cette section se focalise sur l'exécution du travail de comparaison entre éléments de base et sur l'interprétation de ses résultats.

La méthode de comparaison utilisée pour identifier les similarités ou les relations entre les éléments de base TOGAF et DDD est basée sur une comparaison de la sémantique et des relations des éléments.

Pour rappel, le métamodèle TOGAF (ou le cadre de concept) définit un ensemble d'éléments qui permettent de capturer, stocker, filtrer, interroger et représenter des concepts d'architecture, de manière à assurer la cohérence, l'exhaustivité et la traçabilité.

La Figure 28 présente l'ensemble des éléments de base TOGAF ainsi que leurs relations :

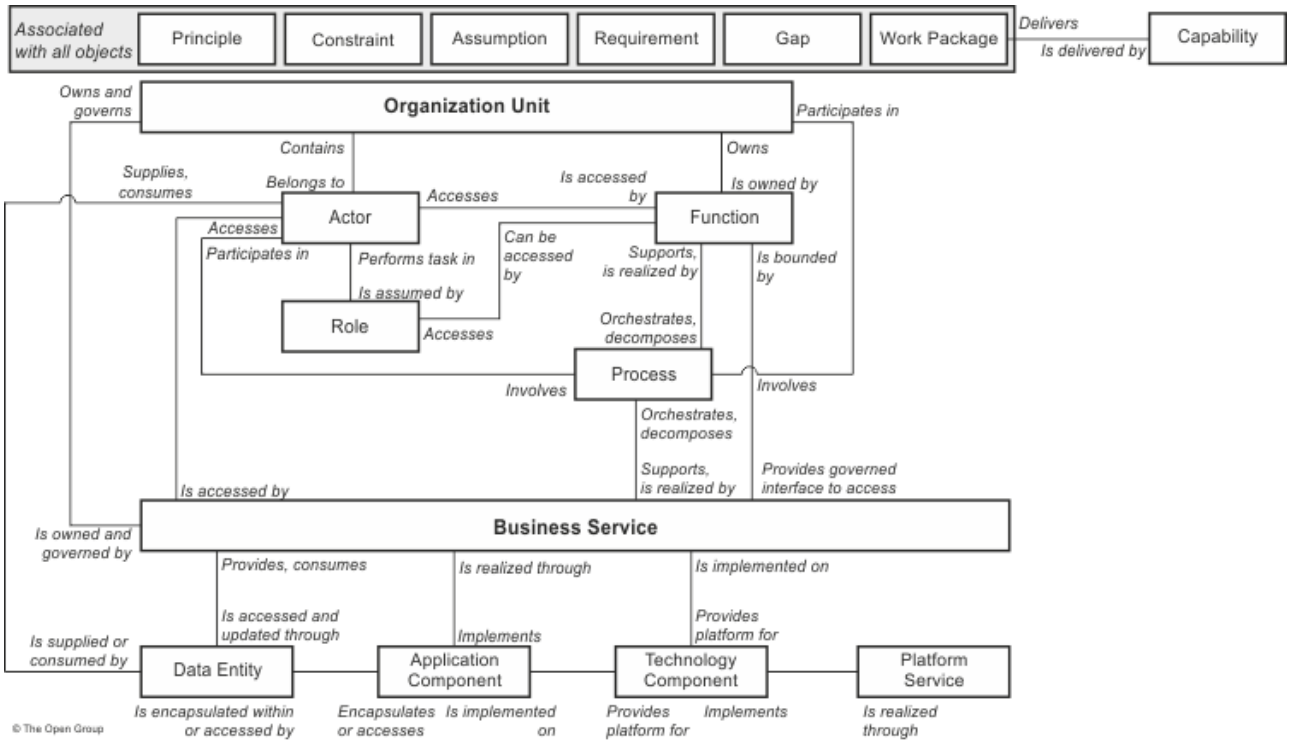

Figure 28 : TOGAF – Métamodèle détaillé des éléments de base de TOGAF [TOGAF9.1]

En complément de la Figure 28, l'annexe 5 de ce mémoire est consacrée à la définition des éléments de base TOGAF. Les informations communiquées dans cette annexe permettent d'approfondir d'avantage la compréhension de la sémantique de ces éléments.

En appliquant la méthode de comparaison citée précédemment, les résultats suivants peuvent être produits (seuls les résultats de comparaison débouchant sur une similitude ou une nouvelle relation sont présentés) :

Comparaison des éléments « Application Service » (DDD) et « Application Component » (TOGAF).

		« Application Service »	« Application Component »
Comparaison sémantique	=	Les deux éléments représentent l'encapsulation d'une fonctionnalité applicative désirée par le métier.	
	≠	Dans l'approche DDD, l'élément est soumis à des considérations spécifiques qui précisent une série de recommandations de conception. Par exemple : l'utilisation d'un verbe du langage omniprésent pour nommer la fonctionnalité, la granularité attendue de la fonctionnalité, ...*.	Néant.
Comparaison des relations	=	Les deux éléments s'appuient sur une manipulation de données pour accomplir leur fonctionnalité.	
	≠	Dans l'approche DDD, l'élément est soumis à des relations spécifiques qui organisent ses relations vis-à-vis des autres éléments de l'extension DDD. Par exemple : l'utilisation de fabriques pour la création d'agrégat, l'utilisation de dépôts pour la mise à jour des agrégats, ….	L'élément est soumis à des relations spécifiques à TOGAF, qui organisent ses relations vis-à-vis des autres éléments de TOGAF. Elles sont nécessaires à l'alignement de la fonctionnalité sur la structure de mise en œuvre. Par exemple : un composant applicatif est l'implémentation d'une fonctionnalité métier, l'exécution d'un composant applicatif est supportée par une infrastructure technique, …
Résultat	?	Sur base de la comparaison, il semble raisonnable d'unifier ces deux éléments. Ils présentent des similarités tant d'un point de vue sémantique qu'au niveau de leurs relations, et aucune spécificité incompatible n'a été identifiée. Les spécificités relevées entre les éléments sont dues à la nature des démarches au sein desquelles ces éléments prennent place. Tel que résumé dans la Figure 29, les adaptations suivantes peuvent être réalisées : **S_1** : Nouvel élément créé est constitué des similarités et de la somme des spécificités propres aux éléments « Application Service » et « Application Component ». **R_1** : Nouvelle relation issue de l'unification des relations : « Data Object » – « Application Service » et « Data Entity » – « Application Component ». **R_2** : Nouvelle relation établie entre « Application Component » et « Domain Service ». De cette façon, l'élément « Application Service » collabore avec des services du domaine pour réaliser sa fonctionnalité. De plus, un élément « Domain Service » représente une spécialisation de l'élément « Application Component ». **R_3** : Nouvelle relation établie entre « Application Component » et « Domain Event ». De cette façon, l'élément « Application Service » acquiert la capacité de traiter les évènements du domaine et de réaliser les opérations de compensations adéquates. **R_4** : Nouvelle relation établie entre « Application Component » et « Factory ». De cette façon, l'élément « Application Service » utilise des fabriques pour créer de nouveaux agrégats. **R_5** : Nouvelle relation établie entre « Application Component » et « Infrastructure Service ». De cette façon, l'élément « Application Service » collabore avec des services d'infrastructure pour réaliser sa fonctionnalité. De plus, un élément « Infrastructure Service » représente une spécialisation de l'élément « Application Component ». **R_6** : Nouvelle relation établie entre « Application Component » et « Repository ». De cette façon, l'élément « Application Service » utilise des dépôts pour récupérer et faire persister des agrégats dans le système de stockage.	

Comparaison des éléments « Data Object » (DDD) et « Data Entity » (TOGAF).

		« Data Object »	« Data Entity »
Comparaison sémantique	=	Les deux éléments représentent de façon atomique le plus petit groupement de données qui exprime un objet métier aux yeux des experts du domaine.	
	≠	Dans l'approche DDD, l'élément est soumis à des considérations spécifiques qui précisent une série de recommandations de conception. Par exemple : l'utilisation d'un verbe du langage omniprésent pour nommer l'objet, ...*.	Néant.
Comparaison des relations	=	Les deux éléments font l'objet d'une manipulation, ceci dans le but d'accomplir une fonctionnalité métier.	
	≠	Dans l'approche DDD, l'élément est soumis à des relations spécifiques qui organisent ses relations vis-à-vis des autres éléments de l'extension DDD. Par exemple : l'encapsulation des objets de données au sein d'agrégats, la spécialisation des objets de donnée en entités ou objets-valeur, ….	L'élément est soumis à des relations spécifiques à TOGAF qui organisent ses relations vis-à-vis des autres éléments de TOGAF et qui sont nécessaires à l'alignement de la fonctionnalité sur la structure de mise en œuvre. Par exemple : une entité de données provient / est consommée par des acteurs métiers, ...
Résultat	?	Sur base de la comparaison, il semble raisonnable d'unifier ces deux éléments. Ils présentent des similarités tant d'un point de vue sémantique qu'au niveau de leurs relations et aucune spécificité incompatible n'a été identifiée. Les spécificités relevées entre les éléments sont dues à la nature des démarches au sein desquelles ces éléments prennent place. Tel que résumé dans la Figure 29, les adaptations suivantes peuvent être réalisées : **S_2** : Nouvel élément créé est constitué des similarités et de la somme des spécificités propres aux éléments « Data Object » et « Data Entity ». **R_7** : Nouvelle relation établie entre « Data Entity » et « Aggregate ». De cette façon, l'élément « Data Entity » est encapsulé au sein d'un agrégat. **R_8** : Nouvelle relation issue de l'unification des relations : « Data Object » – « Application Service » et « Data Entity » – « Application Component ». **R_9** : Nouvelle relation de spécialisation établie entre « Data Entity » et « Entity ». De cette façon, l'élément « Data Entity » peut être spécialisé en Entité pour autant que l'objet ne soit pas défini par ses attributs mais plutôt par une continuité et une identité qui s'étendent dans le temps. **R_{10}** : Nouvelle relation de spécialisation établie entre « Data Entity » et « Value Object ». De cette façon, l'élément « Data Entity » peut être spécialisé en Objet-valeur pour autant que l'objet ne soit pas défini par une identité mais par ses attributs et leurs valeurs.	

En nous basant sur les résultats obtenus suite à la comparaison des éléments de base de DDD et TOGAF, un tableau récapitulatif peut être dressé :

Matrice des similarités et nouvelles relations entre éléments de base DDD et TOGAF	Actor	Application Component	Assumption	Business Service	Capability	Constraint	Data Entity	Function	Gap	Organization Unit	Platform Service	Architecture Principe	Process	Requirement	Role	Technology Component	Work Package
Aggregate							R7										
Application Service		S1					R8										
Bounded Context																	
Data Object		R1					S2										
Domain Service		R2															
Domain Event		R3															
Entity							R9										
Factory		R4															
Infrastructure Service		R5															
Module																	
Repository		R6															
Root Aggregate																	
Value Object							R10										

Figure 29 : Matrice des similarités et nouvelles relations entre éléments de base TOGAF et DDD

Légende :

- S_n : Identifie une unification entre deux éléments
- R_n : Identifie une nouvelle relation entre deux éléments

L'analyse des résultats produits dans le cadre de l'étude comparative des éléments de base DDD et TOGAF permet d'identifier deux possibilités d'unification entre éléments. Les relations des éléments DDD concernés par l'unification sont considérées comme de nouvelles relations applicables aux éléments de base de TOGAF correspondants. La section suivante est consacrée à la production du modèle résultant de ces unifications et nouvelles relations.

Intégration des éléments de base de l'extension DDD au sein du métamodèle TOGAF

Maintenant que le rapprochement entre éléments de base TOGAF et DDD est effectué et que les résultats sont connus, l'intégration des éléments de base de l'extension DDD au sein du métamodèle TOGAF peut être réalisée et représentée sous la forme d'un métamodèle incluant ces deux ensembles d'éléments. Le métamodèle produit est un métamodèle exprimant sans aucune régression les éléments de base de TOGAF et leurs relations, tout en y incluant de façon complète les éléments de base de l'extension DDD et leurs relations.

La Figure 30 présente le métamodèle résultant de l'intégration des éléments de base TOGAF et DDD :

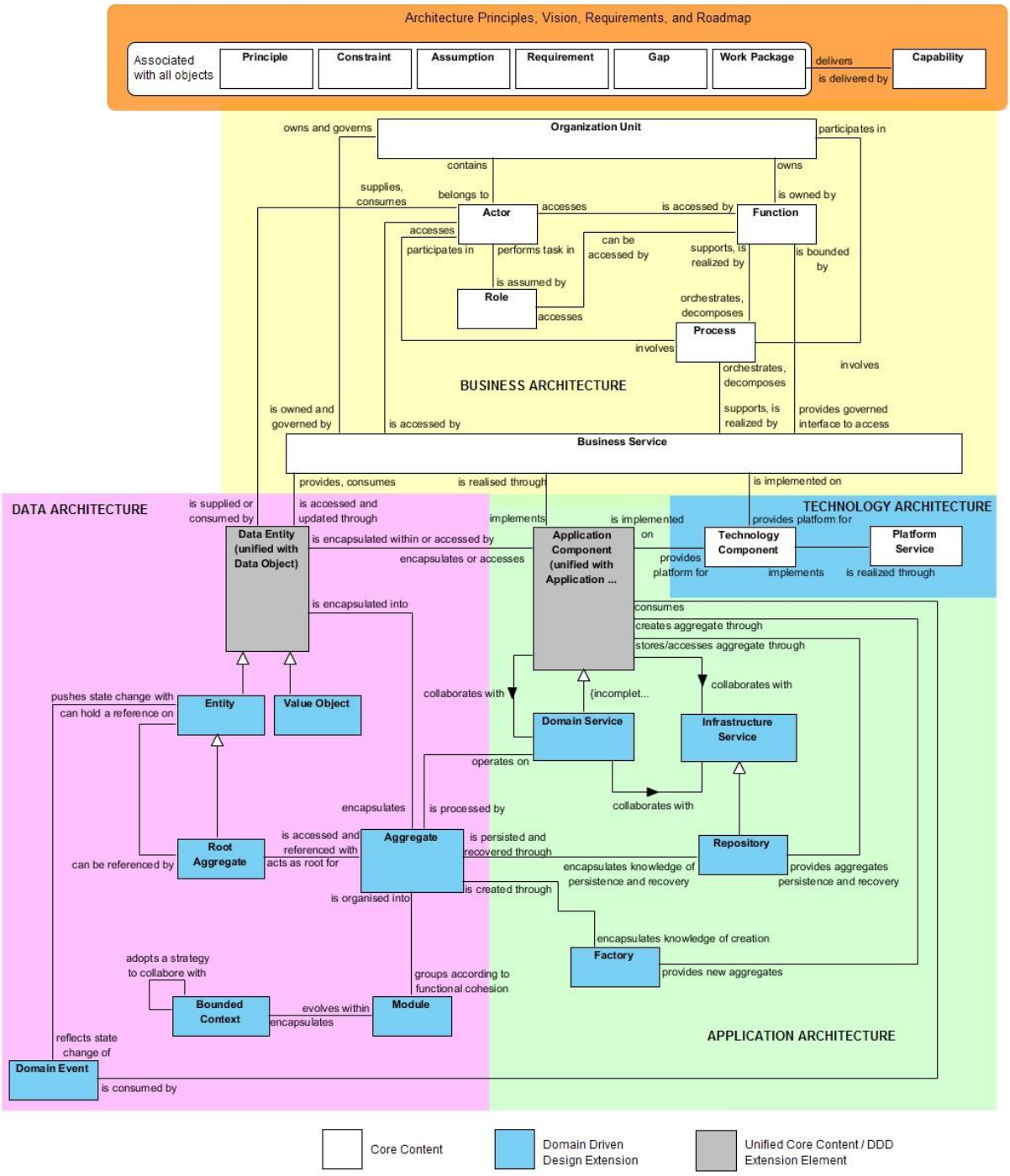

Figure 30 : Métamodèle des éléments de base de TOGAF et de l'extension DDD

L'union du couple « Data Entity » et « Data Object », et l'union du couple « Application Component » et « Application Service » nécessite de procéder à une redéfinition de ces éléments afin d'en exprimer les aspects communs et d'additionner leurs spécificités. Les définitions des autres éléments de base ne sont pas impactées par l'unification des concepts TOGAF et DDD, et restent donc licites.

L'élément de base « Application Component » de TOGAF, modifié suite à son unification avec l'élément de base « Application Service » de DDD, se définit désormais comme ceci :

Application Component

Un **composant applicatif** représente l'encapsulation d'une fonctionnalité applicative alignée sur la structure de mise en œuvre. ✚ La fonctionnalité fait référence à un concept du domaine qui n'appartient pas naturellement à un agrégat (c.-à-d. qui n'est pas de sa responsabilité). (voir Aggregate).

implements

Un composant applicatif représente l'implémentation d'une capacité exposée par les services métiers.

encapsulates or accesses

L'accomplissement de la fonctionnalité nécessite généralement une manipulation d'objets de données issus de plusieurs agrégats.

Pour cela, le composant applicatif interagit avec :

✚ collaborates with

✚ collaborates with

✚ creates agregate through

✚ stores/accesses agregate through

- Des services du domaine (voir Domain Service) pour manipuler des agrégats et obtenir des objets de données.
- Des services d'infrastructure (voir Infrastructure Service) permettant d'utiliser des éléments d'infrastructure particuliers (ex : envoi Email).
- Des fabriques (voir Factory) pour créer de nouveaux agrégats.
- Des dépôts (voir Repository) afin de récupérer des agrégats ayant déjà existés et de faire persister de nouveaux agrégats ou des changements d'états d'agrégat.

✚ consumes

Un composant applicatif est capable de traiter des événements du domaine (voir Domain Event) afin d'appliquer les activités de compensation nécessaires en fonction des attributs de l'événement traité.

Un composant applicatif peut être spécialisé en service domaine (voir Domain Service).

Un composant applicatif ne possède pas d'état.

✚ Identifie les modifications réalisées suite à l'unification des éléments.

On constate que l'intégration dans le métamodèle TOGAF de l'élément de base DDD « Application Service » permet à cet élément de contextualiser son action en se référant à des besoins métiers. Ceci en faisant désormais référence à des éléments de base TOGAF appartenant au domaine d'architecture métier. Les relations héritées de l'élément « Application Component » de TOGAF permettent à l'élément « Application Service » de formaliser le « Pourquoi » de la fonctionnalité applicative représentée par cet élément de base DDD.

La définition de l'élément de base « Data Entity » de TOGAF est également redéfinie, suite à son unification avec l'élément de base « Application Service » de DDD, comme ceci :

Dénomination	Description
Data Entity	Une **entité de donnée** est un ensemble de données reconnues comme une chose par un expert du domaine. ✚ Une entité de donnée peut être spécialisée en Entité (voir Entity) ou en objet-valeur (voir Value Object).
✚ encapsulates	Une entité de donnée est encapsulée au sein d'un agrégat (voir Aggregate).
Is supplied or consumed by	Une entité de donnée est fournie ou consommée par un acteur (voir Actor).
Is accessed and updated through	Une entité de donnée est accédée ou mise à jour au travers d'un service métier (voir Business Service).

✚ Identifie les modifications réalisées suite à l'unification des éléments.

Au même titre que l'élément « Application Component », on peut constater que l'intégration dans TOGAF du concept de « Data Object » permet à cet élément de contextualiser son existence. Ceci en faisant maintenant référence à des éléments de base TOGAF appartenant au domaine d'architecture métier. Les relations de l'élément « Application Component » héritées de TOGAF permettent désormais d'identifier les acteurs métiers concernés dans par la fourniture et la consommation des données.

Identification des similarités et des relations entre éléments de base DDD et éléments d'extension TOGAF

En plus du métamodèle de base, TOGAF propose une collection de six extensions pour ce dernier. Ces extensions sont utilisées pour supporter des décisions d'architecture particulières, au même titre que l'extension DDD qui supporte l'adoption du style d'architecture DDD. Il est plus que probable que des démarches d'architectures, désirant utiliser l'extension DDD, souhaitent également utiliser une ou plusieurs des extensions proposées par TOGAF. Il est donc nécessaire de définir l'intégration des éléments de base de l'extension DDD au sein du métamodèle TOGAF accompagné de ses extensions, ceci afin de couvrir ces cas d'application.

Cette section se concentre donc naturellement sur l'intégration des éléments de base de l'extension DDD au sein des extensions du métamodèle TOGAF. Premièrement, sur la définition de la méthode de comparaison utilisée pour identifier les similarités et relations pouvant être établies entre les éléments. Et deuxièmement, sur l'exécution du travail de comparaison entre ces éléments et sur l'interprétation de ses résultats.

La Figure 31 présente l'ensemble des éléments de base TOGAF et de ses six extensions, ainsi que les liens que ces éléments entretiennent entre eux :

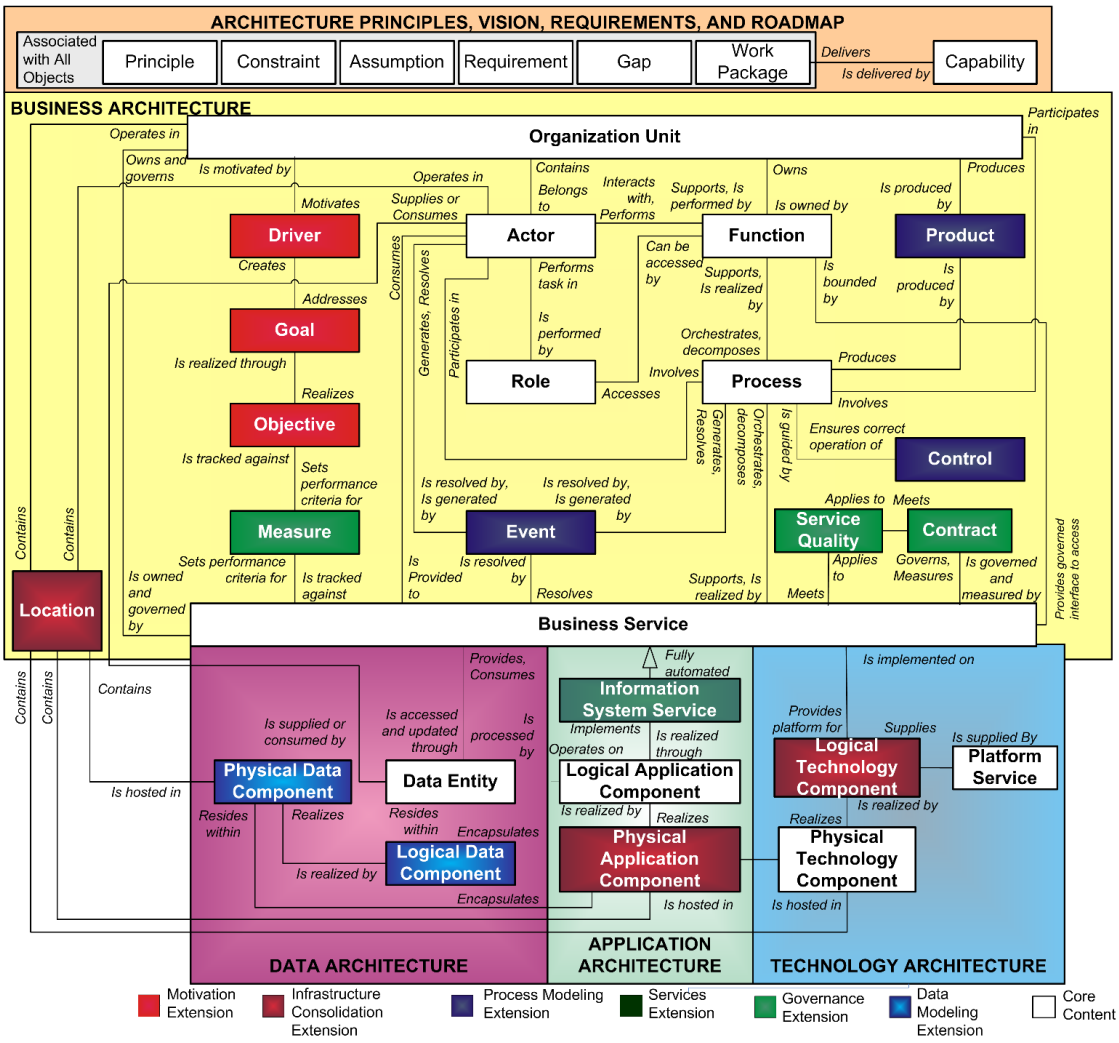

Figure 31 : TOGAF – Métamodèle détaillé des éléments de base de TOGAF et de ses extensions [TOGAF9.1]

En complément de la Figure 31, l'annexe 6 de ce mémoire est consacrée à la définition des éléments d'extensions TOGAF. Les informations communiquées dans cette annexe permettent d'approfondir d'avantage la compréhension de la sémantique de ces éléments.

En appliquant la méthode de comparaison citée précédemment, les résultats suivants peuvent être produits (seuls les résultats de comparaison débouchant sur une similitude ou une nouvelle relation sont présentés) :

Comparaison des éléments « Domain Event » (DDD) et « EVENT» (TOGAF, Process Modeling Extension).

		« Domain Event »	« Event »
Comparaison sémantique	=	Les deux éléments représentent un changement d'état jugé pertinent par les experts du domaine.	
		Dans l'approche DDD, l'élément est soumis à des considérations spécifiques qui précisent une série de recommandations de conception. Par exemple : l'utilisation d'un verbe du langage omniprésent pour nommer l'état, ...*.	Néant.
	≠	Une différence importante concerne la portée des évènements considérés par ces deux éléments :	
		L'élément « Domain Event » identifie un changement d'état concernant une ou plusieurs entités du domaine.	L'élément « Event » identifie un changement d'état concernant n'importe quel élément de l'entreprise.
Comparaison des relations	=	Les deux éléments identifient l'acteur responsable du changement d'état.	
	≠	L'élément est soumis à des relations spécifiques à l'approche DDD qui organisent la relation de l'élément vis-à-vis des autres éléments de l'extension DDD. Par exemple : le traitement des évènements est réalisé par les services applicatifs, ….	L'élément est soumis à des relations spécifiques à TOGAF qui organisent la relation de l'élément vis-à-vis des autres éléments de TOGAF. Par exemple : cet élément est généré/consommé par un processus particulier.
Résultat	?	Sur base de la comparaison réalisée, il ne semble pas raisonnable d'unifier ces deux éléments. En effet, l'analyse de leurs sémantiques laisse apparaître une différence fondamentale en ce qui concerne la portée des évènements représentés. Alors que TOGAF propose un évènement englobant les changements d'état de n'importe quel élément de l'entreprise, DDD propose un évènement du domaine ne concernant que les changements d'états des entités du domaine. Cette différence de portée des évènements empêche l'unification de ces éléments mais laisse tout de même entrevoir la possibilité d'une nouvelle relation pouvant être exprimée entre ces deux éléments. En effet, nous pouvons considérer que les évènements du domaine propres à DDD sont la traduction au sein du domaine d'un sous-ensemble d'évènements capturés par TOGAF. La nouvelle relation créée permettra de faire correspondre un évènement du domaine à un évènement TOGAF en lien avec un ou plusieurs processus étudié dans le cadre de l'architecture métier. Tel que résumé dans la Figure 32, les adaptations suivantes peuvent être réalisées : $\mathbf{R_{11}}$: Nouvelle relation établie entre « Domaine Event » et « Event ». De cette façon, un évènement du domaine est référencé à un évènement TOGAF relatif à un processus métier. Cette relation permet de documenter le contexte métier ayant donné lieu à la création d'un évènement du domaine.	

En nous basant sur les résultats obtenus suite à la comparaison des éléments de base de DDD et des éléments d'extension de TOGAF, un tableau récapitulatif de la comparaison peut être dressé :

Matrice des similarités et des nouvelles relations entre éléments de base DDD et éléments d'extension TOGAF	Contract	Control	Driver	Event	Goal	Information System Service	Location	Logical Data Component	Logical Technology Component	Mesure	Objective	Physical Application Component	Physical Data Compoenent	Physical Technology Component	Product	Requirement
Aggregate																
Application Service																
Bounded Context																
Data Object																
Domain Service																
Domain Event				R11												
Entity																
Factory																
Infrastructure Service																
Module																
Repository																
Root Aggregate																
Value Object																

Figure 32 : Matrice des similarités et nouvelles relations entre éléments d'extension TOGAF et DDD

Légende :

- S_n : Identifie une unification entre deux éléments
- R_n : Identifie une nouvelle relation entre deux éléments

L'analyse des résultats produits dans le cadre de l'étude comparative des éléments de base DDD et des extensions de TOGAF permet d'identifier une nouvelle relation permettant de faire correspondre un évènement du domaine à un évènement métier issu de l'extension spécifique à l'étude des processus métier de l'entreprise.

Intégration des éléments de base DDD au sein des extensions du métamodèle TOGAF

Maintenant que le rapprochement entre éléments d'extension TOGAF et éléments de base de DDD est réalisé, et que les résultats sont connus, l'intégration des éléments de base de DDD au sein du métamodèle TOGAF et de ses extensions peut être réalisée. Le résultat de l'intégration est représenté sous la forme d'une métamodèle incluant ces deux ensembles d'éléments. Le métamodèle produit est un métamodèle exprimant sans aucune régression les éléments de base de TOGAF et de ses extensions, tout en y incluant de façon complète les éléments de base de l'extension DDD.

La Figure 33 présente le métamodèle résultant de l'intégration réalisée :

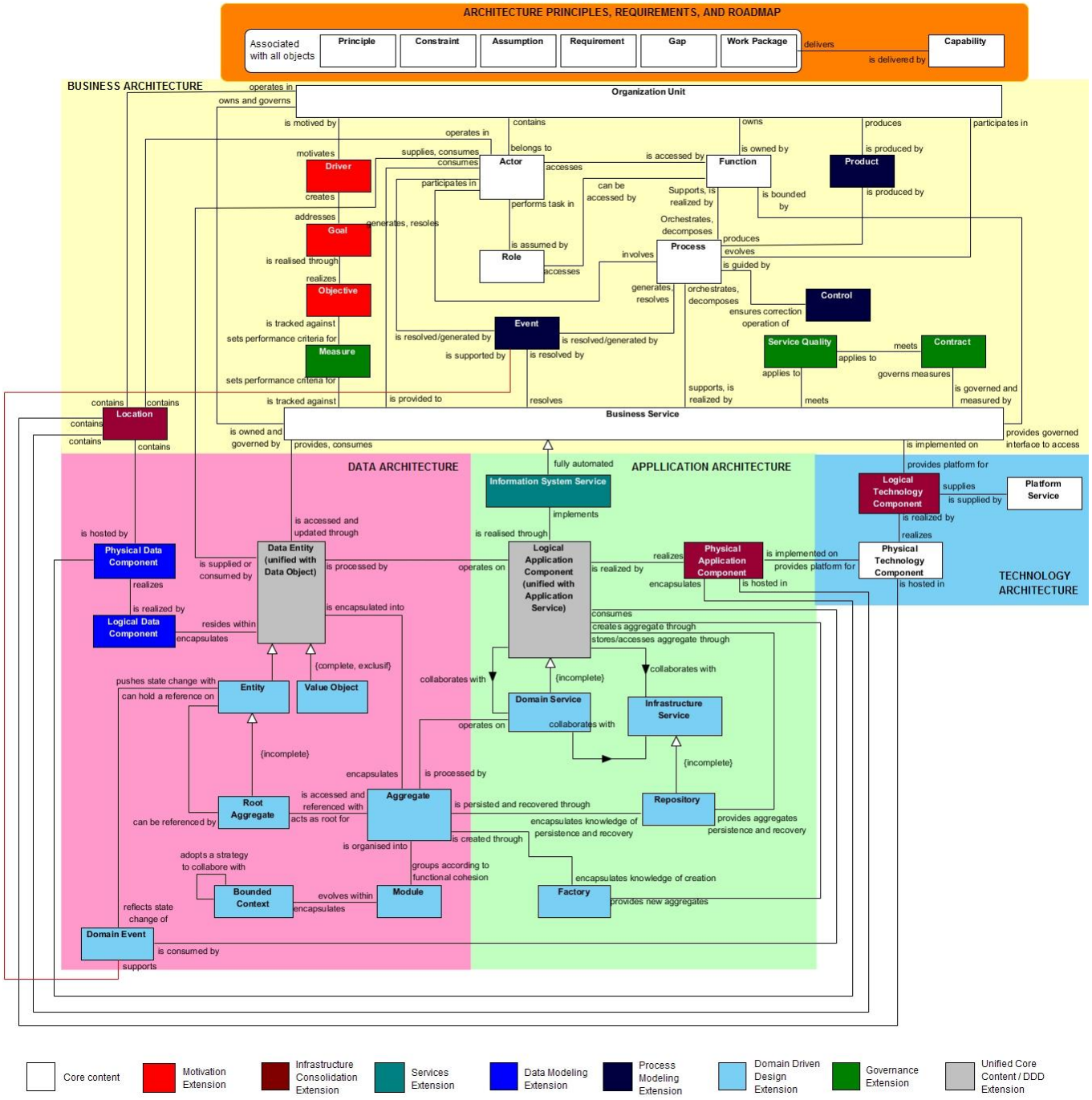

Figure 33 : Métamodèle des éléments de base et d'extension de TOGAF et de DDD

Les définitions des éléments « Event » de TOGAF et « Domain Event » de DDD doivent être reprécisées en tenant compte de la nouvelle relation crée entre ces deux éléments.

Dénomination	Description
Domain Event	Un **évènement du domaine** est une représentation d'un changement d'état d'une ou plusieurs entités (voir Entity) et qui relève potentiellement un intérêt particulier de la part d'autres éléments du domaine. Les informations minimum nécessaires relatives au changement d'état sont : • Heure de survenance • Identités des entités concernées • Nature / Lieu
reflects state change of	Un évènement du domaine communique toute l'information nécessaire afin d'en permettre une interprétation correcte du changement d'état concerné.
is consumed by	Un évènement du domaine est consommé par les services applicatifs (voir Application Service) intéressés.
✚ supports	Un évènement du domaine représente la transcription au sein du domaine d'un évènement identifié dans la cadre de l'étude des processus de l'entreprise. Un évènement du domaine est immutable (car il représente un élément du passé).

On constate que la nouvelle relation établie entre l'élément « Domaine Event » et « Event » permet à l'élément de base de DDD d'appuyer son existence et sa nécessité au travers d'un évènement métier créer dans le cadre de l'étude des processus de l'entreprise. Tous les évènements métier ne donnent pas nécessairement lieu à un évènement du domaine mais tous les évènements du domaine appuient leur existance sur des évènements métier.

Dénomination	Description
Event	Un changement de l'état d'une organisation qui déclenche un traitement. Les événements peuvent être générés à l'intérieur ou à l'extérieur de l'organisation et peuvent être résolus à l'intérieur ou à l'extérieur de l'organisation.
✚ is supported by	Un évènement peut faire l'objet d'une transcription au sein du domaine sous la forme d'un évènement du domaine, dans le cas ou l'évènement concerne un changement d'état d'une ou plusieurs entités (voir Entity).

On constate que la nouvelle relation établie entre l'élément « Domaine Event » et « Event » permet à l'élément de base de TOGAF de pouvoir s'exprimer dans le domaine sous la forme d'évènement du domaine pour autant que l'évènement concerne un changement d'état d'une ou plusieurs entités du domaine.

Identification des nouveaux Artefacts d'architecture

Les artefacts architecturaux traitent en détail de la façon dont le métamodèle de contenu peut être utilisé pour présenter un ensemble de catalogues, de matrices et de diagrammes, ceci afin de répondre aux préoccupations des parties prenantes. TOGAF propose dans son cadre de vues une collection d'artefacts d'architecture produits durant l'exécution des phases du cycle ADM.

Afin de compléter l'adaptation du cadre de contenu dans le cadre démarche d'intégration du style d'architecture DDD au sein de TOGAF : il convient d'adapter le cadre de vue afin d'adapter certains artefacts d'architecture existants et d'y inclure les nouveaux artefacts d'architecture nécessaires permettant de satisfaire les parties prenantes spécifiques à l'approche DDD.

Une vue d'ensemble des nouveaux artefacts d'architecture et des artefacts d'architecture adaptés pour répondre à l'intégration de l'extension DDD dans TOGAF sont présentés à l'aide de la Figure 34. Ce tableau est organisé en regard des phases du cycle ADM concernées par la production de ces artefacts d'architecture.

Phase	Artefact d'architecture	Impact de l'intégration de DDD
Préliminaire	Principles Catalog	1
A : Vision de l'architecture	Stakeholder Map Matrix	1
	Value Chain Diagram	0
	Solution Concept Diagram	0
	Driver/Goal/Objectives Diagram	0
	Driver/Goal/Objectives Catalog	0
	Requirements Catalog	0
B : Architecture métier	Organization/Actor Catalog	0
	Process/Event/Control/Product Catalog	0
	Event Diagram	0
	Role Catalog	0
	Business Service/Function Catalog	0
	Business Interaction Matrix	0
	Actor/Role Matrix	0
	Business Footprint Diagram	0
	Business Service/Information Diagram	0
	Functional Decomposition Diagram	0
	Product Lifecycle Diagram	0
C_1 : Architecture du système d'information (Architecture des données)	Context Map	2
	Bounded Context / Module	2
	Module / Aggregate Matrix	2
	Data Entity/Data Component Catalog	1
	Application/Data Matrix	1
	Conceptual Data Diagram	1
	Logical Data Diagram	1
	Data Dissemination Diagram	0

Phase	Artefact d'architecture	Impact de l'intégration de DDD
C_2 : Architecture du système d'information (Architecture applicative)	Domain Event / Event Matrix	2
	Application Portfolio Catalog	0
	Interface Catalog	0
	Application / Organization Matrix	0
	Role / Application Matrix	0
	Application/Function Matrix	0
	Application Interaction Matrix	0
	Application Communication Diagram	1
	Application and User Location Diagram	0
	Application Use-Case Diagram	1
	Enterprise Manageability Diagram	0
	Process/Application Realization Diagram	1
	Software Engineering Diagram	0
	Application Migration Diagram	0
	Software Distribution Diagram	0
D : Architecture technologique	Technology Standards Catalog	0
	Technology Portfolio Catalog	0
	Application/Technology Matrix	0
	Environments and Locations Diagram	0
	Platform Decomposition Diagram	0
	Processing Diagram	0
	Networked Computing/Hardware Diagram	0
	Communications Engineering Diagram	0

Figure 34 : Niveau d'impact de l'intégration de DDD sur les artefacts d'architecture TOGAF

Légende :

- 0 : Aucun impact
- 1 : Adaptation d'un artefact d'architecture existant
- 2 : Nouvel artefact d'architecture

Stakeholder Map Matrix:

Dans le cadre des recommandations spécifiques à DDD, il est nécessaire de définir un niveau d'intérêt relativement élevé des acteurs relatifs à la conception logicielle vis-à-vis des travaux d'architecture. Ceci doit permettre une compréhension et une évaluation de l'implémentabilité des modèles du domaine, en regard des contraintes conceptuelles ou techniques auquel sont soumis ces acteurs. Typiquement, les participants suivants seront d'avantage consultés dans le cadre de l'étude du domaine métier : Responsable Métier, Expert métier, Concepteur Logiciel, Architecte Technique.

Context Map

La carte de contexte est un nouvel artefact d'architecture attendu à la sortie de la phase C suite à l'intégration de l'extension DDD au sein de TOGAF. La carte de contexte exprime les différents contextes bornés et leurs dépendances ainsi que les stratégies de préservation de l'intégrité des modèles sélectionnées pour organiser ces dépendances. La réalisation de cette documentation permet de fournir une vision d'ensemble et de faciliter l'intégration des

différents contextes bornés entre eux. Sur la carte de contexte, chaque contexte borné possède un nom qui fait partie de langage omniprésent.

Bounded Context / Module Matrix

La matrice de Contexte borné / Module est un nouvel artefact d'architecture attendu à la sortie de la phase C suite à l'intégration de l'extension DDD au sein de TOGAF. La matrice de contexte borné / Module peut être déduite des liens d'encapsulation établis entre les modules et les contextes bornés. Cette matrice permet de donner une vision synthétique de l'organisation et du contenu des différents contextes bornés identifiés, ainsi que des modules dont ils sont composés.

Module / Aggregate Matrix

La matrice de Module / Agrégat est un nouvel artefact d'architecture attendu à la sortie de la phase C suite à l'intégration de l'extension DDD au sein de TOGAF. La matrice de Module / Agrégat peut être déduite des regroupements fonctionnels des agrégats au sein des modules. Cette matrice permet de donner une vision synthétique de l'organisation et du contenu des différents Modules identifiés, ainsi que des agrégats dont ils sont composés.

Data Entity / Data Component Catalog

La matrice entité de données / fonction métier a pour objet de décrire les relations entre les entités de données et les fonctions métier de l'entreprise. Suite à l'intégration de l'extension DDD au sein de la démarche TOGAF, il parait judicieux de renseigner l'agrégat au sein duquel évoluent les données reliées aux fonctions métier. Cette nouvelle information permettra d'identifier les agrégats à manipuler afin de satisfaire une fonction métier particulière. Cette information guidera l'attribution de la gouvernance des agrégats de données aux organisations et facilitera le développement de programmes de gouvernance de données dans l'entreprise.

Application / Data Matrix

La matrice application / données est une table bidimensionnelle avec les composants d'application logique sur un axe et les entités de données sur l'autre axe. Le but de la matrice Application / Data est de décrire la relation entre les applications (c'est-à-dire les composants de l'application) et les entités de données accédées et/ou mises à jour. Suite à l'intégration de l'extension DDD : les données ne doivent plus être directement manipulées par les composants applicatifs mais ce sont les agrégats contenant ces données qui doivent faire l'objet des manipulations. Compte tenu de cette évolution de principe, il parait judicieux de compléter l'information relative aux entités concernées par les manipulations, ceci en indiquant les agrégats desquels les données sont issues. Cette modification permettra d'attribuer l'accès aux agrégats à des applications spécifiques dans l'entreprise et de comprendre où les agrégats sont mis à jour et par quelles applications.

Conceptual Data Diagram

Le but principal du diagramme conceptuel des données est de décrire les relations entre les entités de données dans l'entreprise. Ce schéma est développé pour répondre aux préoccupations des acteurs métier. Il convient de faire évoluer ce diagramme afin d'y inclure certaines considérations faisant suite à l'intégration de l'extension DDD. Les modifications apportées au contenu de ce diagramme sont :

- Les entités de données n'entretiennent plus de dépendances directes entre elles, les dépendances entre entités de données doivent être exprimées sous la forme de dépendances entre entité et racine d'agrégat.
- Les entités de données sont spécialisées en entité / objet-valeur / racine d'agrégat.
- Les entités de données sont encapsulées dans des agrégats.
- Les agrégats sont regroupés en modules selon leur cohésion fonctionnelle.
- Les modules évoluent au sein de contextes bornés.
- Les entités peuvent faire l'objet d'évènements du domaine.

Logical Data Diagram

Le but principal du diagramme logique des données est de montrer les relations entre les entités de données. Ce diagramme est développé pour répondre aux préoccupations des développeurs d'applications et des concepteurs de bases de données. Ce diagramme est une vue plus détaillée du diagramme conceptuel des données abordé dans le point précédent. Il s'appuie donc sur les nouvelles considérations faisant suite à l'intégration de l'extension DDD. La communication de ce diagramme aux différents acteurs des équipes de conception logicielle est très importante, ceci afin de garantir de sa bonne compréhension et de son implémentabilité.

Domain Event / Event Matrix

La matrice évènement du domaine / évènement (de processus) est un nouvel artefact d'architecture attendu à la sortie de la phase C, ceci suite à l'intégration de l'extension DDD au sein de TOGAF. Le but principal de cette matrice est d'identifier les événements du métier (identifiés en Phase B) qui feront l'objet d'une modélisation au sein du domaine sous la forme d'évènements du domaine. Chaque évènement de processus relatif à un changement d'état d'une ou plusieurs entités et qui relève potentiellement un intérêt particulier de la part d'autres éléments du domaine doit faire l'objet d'un évènement du domaine. De cette façon, chacun de ces évènements sera représenté en tant qu'objet du domaine et deviendra ainsi une partie intégrante du modèle. Ce diagramme peut être déduit des relations exprimées entre évènement du domaine et évènement (de processus).

Application Communication Diagram

L'objectif du schéma de communication inter-application est de représenter les interconnections et les communications entre les applications et composants applicatifs du système. Suite à l'adoption de l'extension DDD au sein du métamodèle TOGAF, les

composants applicatifs peuvent maintenant communiquer avec d'autres composants applicatifs spécialisés (services du domaine, services d'infrastructure, dépôts, …).

Application Use-Case Diagram

Un schéma de cas d'utilisation d'application exprime les relations entre les consommateurs et les fournisseurs de services applicatifs. Les services applicatifs sont consommés par des acteurs ou d'autres services applicatifs, ce diagramme permet de définir la mise en œuvre des fonctionnalités de l'application en illustrant comment et quand les fonctionnalités sont utilisées. Suite à l'adoption de l'extension DDD du métamodèle TOGAF, les composants applicatifs peuvent maintenant collaborer avec d'autres composants applicatifs spécialisés (services du domaine, services d'infrastructure, dépôts, …).

Enterprise Manageability Diagram

Evolution identique à l'artefact d'architecture : Application Communication Diagram.

Process/Application Realization Diagram

L'objectif du diagramme processus / réalisation d'application est de décrire clairement la séquence d'événements lorsque plusieurs composants applicatifs sont impliqués dans l'exécution d'un processus métier. Suite à l'adoption de l'extension DDD du métamodèle TOGAF, ce diagramme pourra mettre en œuvre un séquençage d'opérations encapsulées par des services applicatifs spécialisés, et de cette façon représenter la granularité et la ventilation des responsabilités fonctionnelles au sein des composants applicatifs.

Inspection du cadre de capacité

TOGAF fournit un cadre de capacité, ceci afin de permettre de guider l'entreprise dans la mise en place de ce qui est nécessaire afin que l'entreprise se mette en capacité de supporter la démarche d'architecture d'entreprise désirée.

Une des tâches de l'architecte est d'adapter le cadre de capacité en tenant compte des spécificités de l'entreprise et de la démarche d'architecture, ceci y compris en tenant compte des principes d'architecture définis.

L'inspection du cadre de capacité est réalisée en examinant les ressources, les recommandations, et les modèles fournis par TOGAF afin d'y déceler les éléments qui soutiennent l'approche DDD et ceux qui devraient être adaptés.

Après examen, il semble que le cadre de capacité ne doivent pas faire l'objet d'une adaptation pour mieux soutenir l'intégration du style d'architecture DDD. Ceci s'explique assez simplement : les points d'attention et de considération concernés par le cadre de capacité sont fortement éloignés des points de considérations relatifs au style d'architecture DDD.

L'exécution de la mise en place de la Capacité d'Architecture nécessitera de disposer de ressources nécessaires spécifiques à l'approche DDD. Mais ceci ne sera finalement que le résultat de l'exécution de l'étape traitant de la définition des capacités requises pour supporter la réalisation de l'architecture d'entreprise.

2.1.3 Etape : Alimenter le référentiel d'architecture

Pour rappel, la phase préliminaire de TOGAF comprend la création d'un dépôt d'architecture avec une collection initiale de matériel. Le dépôt d'architecture d'entreprise contient une collection de modèles, patterns, de descriptions et d'autres artefacts permettant de guider le développement de l'architecture. Ces éléments peuvent résulter de travaux antérieurs ou provenir de bonnes pratiques, de techniques et de concepts issus du monde de l'informatique et répondant à des besoins particuliers.

Le guide de Reference de DDD (voir [DDDRef, 2015]) et le livre d'Eric Evans en la matière (voir [DDD, 2004]) compilent de nombreux matériaux susceptibles d'être pertinents dans le dépôt initial du dépôt d'architecture. Ces ouvrages décrivent l'architecture de référence du style d'architecture DDD, ce qui constitue une structure logique sous-jacente soutenant le développement et l'évaluation d'architectures conçues et construites en utilisant les principes et les concepts informatiques traditionnels et orientés DDD.

Il conviendra donc, dans le cadre d'une intégration du style d'architecture DDD à la démarche TOGAF, d'inclure dans le dépôt d'architecture tous les éléments nécessaires à la soutenance de la démarche DDD. Ceci inclut :

- Les patterns sélectionnés pour implémenter les différents concepts issus de l'approche DDD, et plus particulièrement son approche de conception dirigée par le modèle.

- L'architecture de référence de DDD, qui donne un aperçu des quatre couches de l'architecture de référence. L'ouvrage d'Eric Evans apporte des exemples et justifications décrivant les principales responsabilités des couches et de leurs éléments constitutifs.

En résumé, lors de la réalisation de la phase préliminaire du cycle ADM de TOGAF, un certains nombre d'éléments doivent être pris en compte afin d'adapter la capacité d'architecture, et de cette façon lui permettre de soutenir la mise en œuvre du style d'architecture DDD. Cette phase d'adaptation inclut :

- La définition et l'intégration des principes DDD dans le catalogue de principes.
- La définition de l'extension DDD.
- L'intégration de l'extension DDD au sein du métamodèle TOGAF, y comprit de ses extensions.
- La définition des artefacts d'architecture adaptés ou nouvellement créés pour exprimer les points de vue spécifiques à l'extension DDD.
- L'alimentation du référentiel d'architecture avec une collection de modèles, patterns, … permettant de guider le développement de l'architecture au style DDD.

Les phases A, B, C, D, E, F, G, H et de gestion des exigences sont, quant à elles, plutôt considérées comme des phases d'exécution de la Capacité d'Architecture définie durant la phase préliminaire. Les exécutions de ces phases n'apporteront pas d'adaptations supplémentaires de TOGAF pour supporter le style d'architecture DDD mais seront simplement l'exécution de l'adaptation réalisée durant la phase préliminaire.

Cependant, bien que des adaptations supplémentaires ne semblent pas nécessaires pour permettre à TOGAF de supporter l'approche DDD, il est important de garder à l'esprit que les principes DDD constituent plus qu'un simple amalgame de recommandations et de techniques. Au-delà de ses considérations de conceptions, il faut percevoir le style d'architecture DDD comme un état d'esprit, une manière de percevoir et de traiter le domaine métier. L'approche DDD et ses enjeux doivent être clairement comprit par tous les acteurs de l'architecture d'entreprise mais également par les acteurs de la conception logicielle, ceci afin de permettre la création d'une culture d'entreprise englobant ses principes et sa façon de raisonner à propos de domaine. Il est donc indéniable que les principes DDD et son adoption par l'organisation influenceront les travaux réalisés, au-delà des adaptations proposées dans le cadre de ce mémoire, citons par exemple :

- La prise en compte des contextes bornés dans le cadre de la segmentation de l'entreprise réalisée en phase A : Vision.
- L'exécution des travaux d'architecture métier réalisé en phase B, devant tenir compte des principes de distillation du domaine métier.
- La planification de la migration réalisée en phase H, devant tenir compte des dépendances entre contextes bornés afin d'initier les projets.
- …

Signalons qu'étant donné le manque de cas d'application concret de l'extension DDD définie dans le cadre de ce mémoire, il probable que des ajustements doivent encore s'opérer afin d'affiner et de consolider sa mise en œuvre. Il n'est donc pas exclu, dans un premier temps, de devoir adapter la capacité d'architecture durant l'exécution du cycle ADM, afin de tenir compte des ajustements jugés nécessaires. Cela afin de rendre encore plus pertinente l'intégration de DDD au sein de TOGAF. Il conviendra alors dans ce cas de revoir les travaux d'architectures déjà réalisés, afin de les adapter suivant les modifications apportées à la capacité d'architecture.

Conclusion

En génie logiciel, l'importance de l'alignement entre la stratégie d'entreprise le système d'information est capital. Pour comprendre et raisonner à propos de son système d'information, l'entreprise doit avoir une vision claire de son domaine métier qui devient de plus en plus complexe et morcelé aux travers de différents applicatifs isolés. Même si l'industrialisation et l'informatisation des processus métier est chose courante de nos jours, il reste difficile de maîtriser l'évolution du parc applicatif et de laisser les éléments métier au cœur des différents processus, logiciels et outils proposés.

Le premier objectif de ce mémoire est de mieux comprendre ce qu'est une architecture d'entreprise ainsi que de se faire une idée sur le style d'architecture DDD et les problématiques que ces démarches solutionnent. Nous avons pu comprendre, au travers des différents chapitres, ce qu'est l'architecture d'entreprise et plus particulièrement le cadre d'architecture d'entreprise TOGAF, et pourquoi une approche DDD peut-être intéressante pour recentrer le domaine métier au centre des préoccupations du SI.

Le second objectif est de vérifier que TOGAF est bien compatible avec un style d'architecture pour lequel il ne propose pas, nativement, d'extension. Le but recherché est de montrer dans quelle mesure TOGAF peut tenir compte des principes du style d'architecture DDD afin de proposer une méthode unifiée permettant l'application conjointe de ces deux approches. A ma connaissance, ce genre d'approche n'a encore jamais été proposé. Ce mémoire apporte donc une première contribution dans ce domaine en proposant une évaluation de la capacité de TOGAF à soutenir une approche DDD. La capacité d'adaptation de TOGAF, exposée au travers de l'exécution de la phase préliminaire du cycle ADM est l'élément fondamental caractérisant son adaptabilité. En effet, cette étape permet d'adapter le contenu de TOGAF pour faciliter l'intégration d'éléments extérieurs qui ne seraient pas pris en charge nativement par le cadre d'architecture. Cette capacité d'adaptation permet de vérifier que TOGAF dispose des capacités nécessaires à la soutenance du style d'architecture DDD. L'exécution de cette adaptation permettant de vérifier la compatibilité et la complémentarité des paradigmes propres à l'approche DDD et des paradigmes de la démarche TOGAF.

Le troisième objectif est d'adapter TOGAF afin que ce dernier soutienne l'adoption du style d'architecture DDD. Nous avons vu que la première adaptation à faire était d'inclure les principes d'architecture spécifiques à DDD. Ceux-ci représentent la pierre angulaire de la démarche d'intégration. Ensuite, nous avons fait évoluer le cadre de contenu afin que ce dernier puisse intégrer les concepts relatifs à la modélisation dirigée par le modèle. Rappelons également qu'il est important de populer le dépôt d'architecture initiale avec les patterns sélectionnés. La définition des livrables attendus en regard des différentes phases du cycle ADM est également adaptée afin de prendre en compte les nouveaux besoins et les nouvelles parties prenantes qui sont venus s'ajouter suite à la d'intégration de DDD.

Nous pouvons donc conclure que TOGAF utilise sa capacité d'adaptation pour soutenir et permettre l'intégration du style d'architecture DDD. Bien que l'ensemble des éléments spécifiques à l'approche

DDD soient soutenus par TOGAF, cela n'indique pas, pour autant, qu'un autre style d'architecture soit entièrement compatible avec ses principes.

Il est important de signaler que l'application de DDD ne se limite pas au contexte de TOGAF. Les principes et recommandations relatifs à l'approche DDD s'étendent au-delà des frontières de l'architecture d'entreprise. L'application du style d'architecture DDD débute dans la démarche d'architecture d'entreprise et s'étend jusqu'aux projets d'implémentation et de maintenance. Les principes de l'approche DDD s'appliquent jusque dans le code applicatif qui sera écrit ou encore durant les phases de tests menées pour entre autres vérifier l'intégrité des modèles.

Le sujet étant très vaste, il est difficile de donner l'ensemble des pistes de recherches futures.

Cependant, nous pouvons en citer quelques-unes :

- Compléter ce guide en y apportant des exemples concrets issus de l'application de cette approche d'application combinée de TOGAF et DDD.
- Appliquer cette démarche dans plusieurs situations particulières afin de valider qu'elle fonctionne également dans d'autres contextes.
- Formaliser les éléments de base de l'extension DDD sous la forme de plugin à destination d'outils de modélisation.
- …

Références

Bibliographie

- **[AGF, 2012]**
  Open Group, *ArchiMate 2.1 Translation Glossary : English – French (Document Number: C13B)*, Open Group, 2014

- **[DDD, 2004]**
  Eric Evans, *Domaine-Driven Design – Tackling Complexity in the Heart of Sotware,* DDD Series, 2004

- **[DDDRef, 2015]**
  Eric Evans, *Domaine-Driven Design Reference – Definitions and Pattern Summaries,* DDD Series, 2015

- **[DDDViteFait, 2006]**
  Abdek Avram, Floyd Marinescu, *DDD vite fait*, C4Media Inc, 2006

- **[ISO/IEC/IEEE42010, 2011]**
  Institute of Electrical and Electronics Engineers, *INTERNATIONAL STANDARD ISO/IEC/ IEEE 42010: Systems and software engineering — Architecture description Ingénierie des systèmes et des logiciels — Description de l'architecture*, IEEE, 2011

- **[Krutchen, 1999]**
  Philippe Kruchten, *The Rational Unified Process*, Addison-Welsey, 1999

- **[NILES, 2006]**
  Niles E Hewlett, *The USDA Enterprise Architecture Program*, USDA-OCIO, 2006

- **[OGF, 2012]**
  Open Group, *TOGAF 9.1 Translation Glossary : English – French (Document Number: C127)*, Open Group, 2012

- **[Perry&Wolf, 1992]**
  Perry D. E. & Wolf A. L., *Foundations for the study of software architecture*, SIGSOFT Softw. Notes Vol 17 N°4, 1992

- **[TOGAFGuideDePoche, 2010]**
  Andrew Josey et Al, *TOGAF Version 9 Guide de Poche*, Open Group, 2010

Webographie

- **[ArchiMetric]**
  https://www.archimetric.com/what-is-togaf/
  (Consulté le 11/02/2017)

- **[BSPEAHistory]**
  https://www.researchgate.net/publication/308936998_The_History_of_Enterprise_Architecture_An_Evidence-Based_Review
  (Consulté le 20/05/2016)

- **[EADomainsWiki]**
  https://en.wikipedia.org/wiki/Architecture_domain
  (Consulté le 03/09/2016, 04/09/2016)

- **[GartnerGroup]**
  http://www.gartner.com/it-glossary/enterprise-architecture-ea/
  (Consulté le 25/07/2016)

- **[MIT]**
  http://www.iese.edu/en/files/6_29338.pdf
  (Consulté le 25/07/2016)

- **[OxfordArchitecture]**
  https://en.oxforddictionaries.com/definition/architecture
  (Consulté le 25/05/2017)

- **[RealIRM]**
  https://www.realirm.com/enterprise-architecture/architecture-domains
  (Consulté le 04/09/2016)

- **[TOGAF9.1]**
  http://pubs.opengroup.org/architecture/togaf9-doc/arch/
  http://pubs.opengroup.org/architecture/togaf9-doc/arch/toc.html
  (Consulté à partir du 05/07/2016 et durant toute la réalisation de ce mémoire)

- **[USGov]**
  Cette citation est disponible de nombreux endroits, par exemple,
  http://www.cio.ca.gov/ea/
  (Consulté le 25/07/2016)

- **[ZachmanWiki]**
  https://en.wikipedia.org/wiki/Zachman_Framework
  (Consulté le 14/08/2016, 16/08/2016)

Annexes

Annexe 1 – Historique détaillé de l'architecture d'entreprise

L'architecture d'entreprise fait une première et timide apparition dans les années 60 dans les travaux de P. Duane Walker qui est à l'origine d'une première formalisation des documents architecturaux. A la fin des années 60, P. Duane Walker, devenu directeur d'IBM, continua ses recherches relative aux documents de planification de l'architecture, ceci donnera lieu à BSP – Business System Planning (voir Figure 1-A : Méthodologie BSP). Cette nouvelle approche dont IBM fit la promotion se penchait sur des aspects tels que : la concrétisation d'opportunité grâce à la technologie, la production de plans de migration technologique, le support à la décision et le support du développement.

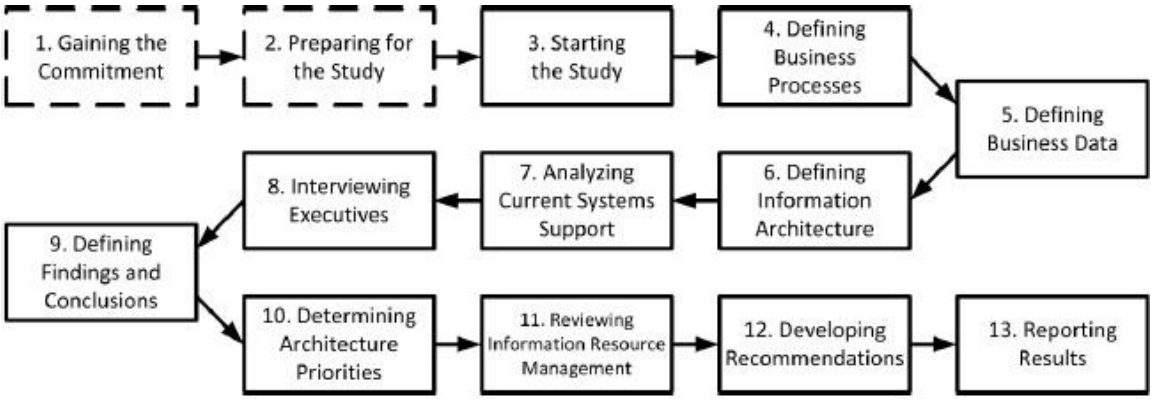

Figure 1-A : Méthodologie BSP[16]

En 1986, un service de recherche créé à la suite du projet de recherche parrainé par un groupe de sociétés (incluant IBM) et visant à trouver des moyens optimaux pour décrire une architecture de systèmes distribués, introduit un cadre d'architecture d'entreprise défini comme une structure logique pour l'organisation de la description d'une entreprise. Le cadre PRISM EA a été le premier cadre EA publié dans ce que l'on considère comme la compréhension moderne de ce concept. Le cadre PRISM EA organise une description architecturale en 16 catégories en fonction de quatre domaines (organisation, données, applications et infrastructure) et quatre types (inventaire, les principes, les modèles et les normes)(voir Figure 1-B : PRISM EA).

[16] https://www.researchgate.net/publication/308936998_The_History_of_Enterprise_Architecture_An_Evidence-Based_Review

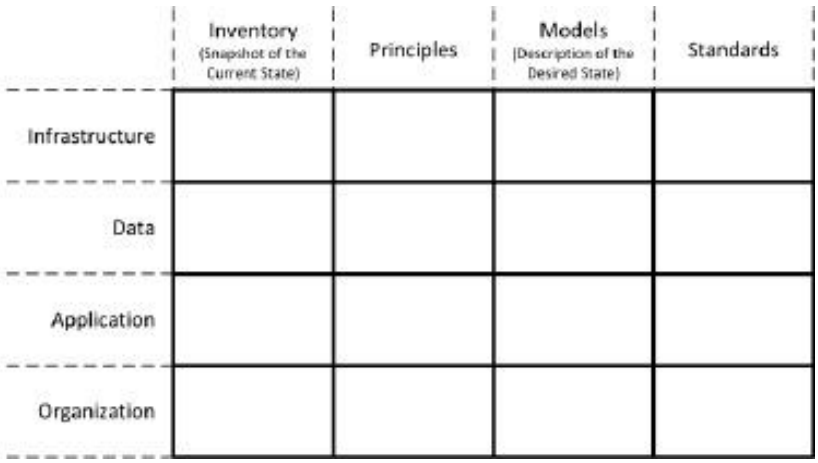

Figure 1-B : PRISM EA[17]

Mais il faudra patienter jusqu'en 1987, avec la publication au Journal IBM Systems d'un article intitulé « Un cadre d'architecture pour les systèmes d'information » pour voir réellement apparaître ce que la majorité des experts en ce domaine référence comme la première démarche d'architecture d'entreprise. Dans ce document J.A. Zachman, ancien élève de P. Duane Walker et employé d'IBM, fait évoluer BSP et pose des bases solides en explicitant les défis et la vision des architectures d'entreprise. L'objectif poursuivi par cet article était de permettre la gestion de la complexité des systèmes de plus en plus distribués. La vision de Zachman étant que la valeur de l'entreprise et de l'agilité pourraient mieux être rencontrés par une approche holistique de l'architecture des systèmes qui exprimait explicitement chaque question importante selon plusieurs perspectives (le Quoi, le Qui, le Ou, le Comment et le Pourquoi). Zachman décrit une approche multi perspectives (#6) aux architectures systèmes, ce qui est à l'origine du cadre architectural des systèmes d'information et bientôt renommé pour être le premier cadre d'entreprise d'architecture (voir Figure 1-C : Le cadre d'AE de Zachman).

	Why	How	What	Who	Where	When
Contextual	Goal List	Process List	Material List	Organisational Unit & Role List	Geographical Locations List	Event List
Conceptual	Goal Relationship	Process Model	Entity Relationship Model	Organisational Unit & Role Relationship Model	Locations Model	Event Model
Logical	Rules Diagram	Process Diagram	Data Model Diagram	Role Relationship Diagram	Locations Diagram	Event Diagram
Physical	Rules Specification	Process Function Specification	Data Entity Specification	Role Specification	Location Specification	Event Specification
Detailed	Rules Details	Process Details	Data Details	Role Details	Location Details	Event Details

Figure 1-C : Le cadre d'AE de Zachman[18]

[17] https://www.researchgate.net/publication/308936998_The_History_of_Enterprise_Architecture_An_Evidence-Based_Review

[18] https://en.wikipedia.org/wiki/Zachman_Framework

En 1994, le département américain de la Défense (DoD) publie un cadre d'architecture pour la gestion de l'information, ce cadre est connu sous la dénomination TAFIM (Technical Architecture Framework for Information Management). La pensée de Zachman semble avoir fortement influencé cette approche.

Parallèlement en 1994, l'Open Group sélectionne TAFIM comme base pour le développement de TOGAF (The Open Group Architecture Framework). La démarche proposée par TOGAF est l'adoption d'une vue stratégique, construite à l'échelle de l'entreprise mais également orientée vers la technologie. L'un de ses principaux objectifs est de permettre la rationalisation des domaines informatiques, le plus souvent désordonnés. La première version de TOGAF verra le jour au début de l'année 1995 en proposant dans un premier temps son cadre d'architecture, complété ultérieurement par son processus de développement (voir Figure 1-D : TOGAF – Cadre et Processus).

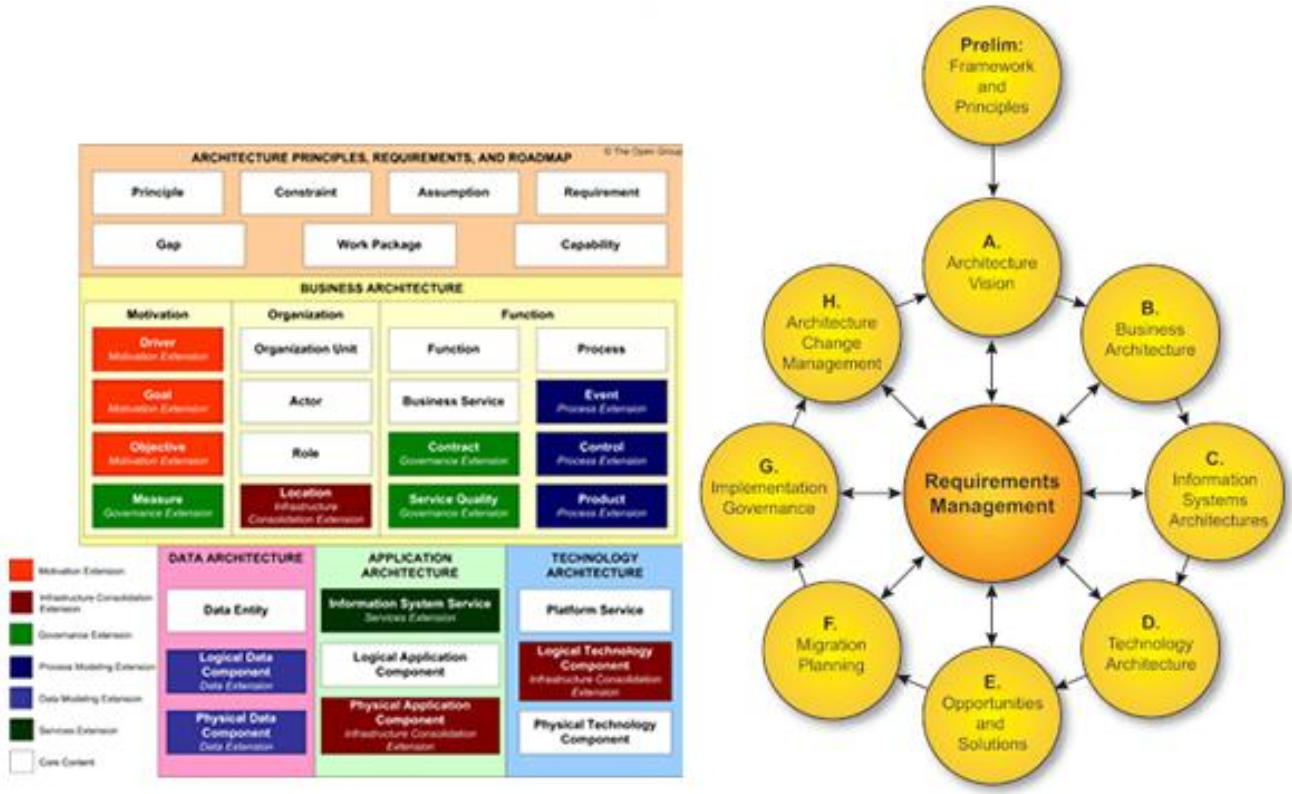

Figure 1-D : TOGAF – Cadre et Processus[19]

En 1998, le Conseil américain fédéral (CIO) initie son premier grand projet du genre, le cadre d'architecture d'entreprise fédérale (FEAF). La version 1.1 de ce cadre a été publiée en Septembre 1999. Le FEAF décrit une approche, y compris les modèles et les définitions, pour développer, documenter et décrire l'architecture pour les segments fonctionnels multi-organisationnelles du gouvernement fédéral. Dans la continuité de l'approche du cadre de Zachman, le FEAF propose des modèles pour décrire le métier, les données, les applications et la technologie. La mise en œuvre du cadre d'architecture FEAF au sein du gouvernement fédéral américain sera imposée par la loi nommée Clinger/Cohen qui conduira à de gros investissements mais qui ne rencontrèrent pas le succès escompté.

[19] http://pubs.opengroup.org/architecture/togaf9-doc/arch/

Au début des années 2000 se fit progressivement un transfert de compétence entre le CIO et le Bureau de la gestion et du budget (OMB) en ce qui concerne l'architecture d'entreprise fédérale. En 2002, l'OMB décide de réévaluer et de renommer la méthodologie FEAF sous la dénomination : Federal Enterprise Architecture (FEA). FEA est la dernière tentative du gouvernement fédéral d'unir ses innombrables agences et fonctions sous une seule architecture d'entreprise commune et omniprésente. Notons qu'à cette époque, FEA en est encore à ses balbutiements, la plupart des pièces majeures de ce cadre n'ayant pas été disponibles avant sa version 1.2 publiée elle en 2006.

En 2005, Gartner décide de prendre des mesures afin de devenir une force dominante de l'AE dans le secteur privé. A cette époque, Gartner était l'une des organisations les plus influentes spécialisées dans le conseil de niveau CIO. Et donc en 2005, Gartner réalisant qu'il ne pouvait pas rivaliser avec Meta Group décida alors d'en faire l'acquisition. Suite à l'achat de Meta Group, Gartner / Meta passa une année à examiner l'expérience d'architecture d'entreprise et les méthodologies en analysant les propositions issues des deux compagnies. Les deux entreprises firent beaucoup d'efforts afin de trouver la meilleure façon de concilier leurs approches, souvent très différentes.

En 2006, le Centre de recherche des systèmes d'information du MIT publie une livre intitulé : "Enterprise Architecture As Strategy"[20]. Cet ouvrage rapporte le travail entamé quelques années plutôt. Les auteurs de cet ouvrage accordent une grande importance à la nécessité pour les architectes d'entreprise de se concentrer sur les processus métier de l'entreprise. La justification est que les entreprises excellent parce qu'elles ont décidé de quels processus métier doivent être soutenus par l'informatique. L'importance de l'engagement des gestionnaires d'entreprises dans la démarche d'architecture d'entreprise est également mise en avant.

Au fil des années suivantes, de multiples recherches et améliorations des cadres conceptuels d'AE existants ont étés menés. Pour aboutir actuellement à une multitude de méthodologie, d'approche et d'outils dont TOGAF semble accaparer le plus d'attention de la part des entreprises.

Le problème de fond pour lequel l'architecture d'entreprise semble apporter une réponse est d'une part l'augmentation des coûts IT et d'autre part une perte de valeur ajoutée de l'IT pour le métier. Ces difficultés, d'abord identifiées il y a des dizaines d'années, ont aujourd'hui atteint un point de crise. Le coût et la complexité des systèmes informatiques ont augmenté de façon exponentielle, de ce fait les chances de tirer la valeur réelle de ces systèmes ont considérablement diminuées. Les grandes organisations ne peuvent plus ignorer cette complexité et ces obstacles. Alors qu'il y a 30 ans, la pratique de l'Architecture d'Entreprise semblait marginale, elle est maintenant considérée comme providentielle !

[20] Jeanne W. Ross, Peter Weill, and David C. Robertson, *Enterprise Architecture as Strategy - Creating a Foundation for Business Execution*, Harvard Business School Press, 2006

Annexe 2 – L'architecture d'entreprise et les instruments de gouvernance

Chapitre sorti de l'état de l'art en vue d'en alléger la lecture, cependant la lecture de cette section peut s'avérer pertinente afin d'objectiver les complémentarités et les points de jonctions de TOGAF vis-à-vis de divers instruments de gouvernance populaires.

La première finalité rencontrée de l'architecture d'entreprise est d'être utilisée comme un instrument de gestion de l'entreprise, autant pour ses opérations quotidiennes que dans le cadre de ses développements futurs. La question légitime que l'on est en droit de se poser concerne le positionnement de l'architecture d'entreprise et la manière dont elle s'intègre aux pratiques et instruments de gestion d'entreprise établis.

Pour décrire le positionnement de l'architecture d'entreprise dans le contexte de la gouvernance d'entreprise, examinons-la dans le cadre d'un certain nombre de pratiques et de normes bien connues en matière de gestion générale :

Management de la stratégie → The Balanced Scorecard (BSC)

BSC est un instrument de gouvernance qui aide une entreprise à clarifier et à mettre en œuvre sa vision et sa stratégie. BSC suggère une visualisation de l'entreprise à partir de 4 points de vue :

- client (image de marque, satisfaction client, …)
- financier (concerne la chaine de valeur de l'entreprise, l'actionnariat, …)
- métier (efficacité et efficience des processus)
- apprentissage et croissance (évolution, amélioration, …)

Pour chacun de ces points de vue BSC adopte une structure en trois couches :

- La mission (par exemple, pour devenir le fournisseur préféré des clients);
- L'objectif (par exemple, de fournir aux clients de nouveaux produits);
- Les mesures (par exemple, le pourcentage du chiffre d'affaires généré par les nouveaux produits).

Si nous examinons le rôle de l'architecture d'entreprise en tant qu'instrument de gestion, il semble que celle-ci soit particulièrement utile dans le cadre du point de vue métier proposé par BSC. En effet, une architecture d'entreprise documente de nombreux paramètres opérationnels et les liens qui existent entre eux. Sur base de cette connaissance, diverses analyses de performances peuvent être réalisées et donc supporter la démarche d'analyse du métier initiée par la méthodologie BSC. Ce n'est pas tout, l'architecture d'entreprise apporte également une plus-value du point de vue de l'apprentissage et de la soutenance de la croissance. La capacité d'anticipation et d'évolution d'une entreprise est primordiale afin de répondre à un environnement changeant. Pour supporter l'agilité de l'organisation, il est important d'évaluer l'impact et la faisabilité des changements envisagés. Les analyses d'impact réalisées dans le cadre d'une architecture d'entreprise semblent être une solution pertinente pour soutenir cette activité.

Exécution de la stratégie → EFQM

L'EFQM (European Foundation for Quality Management) est un autre instrument de gouvernance important. EFQM a une portée beaucoup plus large que la norme ISO 9001, il met l'accent non seulement sur la gestion de la qualité, mais fournit un ensemble de cadres de gestion pour l'excellence de la performance de l'ensemble de l'organisation.

EFQM positionne l'architecture d'entreprise comme un instrument important concernant la politique et la stratégie de l'entreprise. Sur la base de sa mission et de sa vision, une entreprise affichera sa volonté à déterminer les politiques et les stratégies nécessaires pour satisfaire les besoins et attentes de ses parties prenantes, ceci aussi bien pour le présent que pour l'avenir. Une architecture d'entreprise est un instrument précieux dans l'opérationnalisation et la mise en œuvre de ces politiques et stratégies. Tout d'abord, elle permet de mieux comprendre la structure et le fonctionnement de l'entreprise dans son ensemble en créant une vue globale de sa structure, de son organisation, de ses processus, de ses systèmes d'information et de son infrastructure. Une telle documentation semble indispensable lors de la formulation d'une stratégie cohérente. En outre, une architecture d'entreprise contribuera au développement, à la gestion et à la communication des normes de fonctionnement de toute l'entreprise,. Ceci est absolument nécessaire afin d'assurer que les politiques de l'entreprise sont en effet mis en œuvre. Enfin, en fournissant une meilleure compréhension des impacts des changements, l'architecture d'entreprise sera d'une aide précieuse pour la création des feuilles de route pour l'avenir, feuille de route nécessaire pour évaluer et exécuter la stratégie de l'entreprise à long terme.

Management de la qualité → ISO 9001

La certification ISO 9001: 2000 (ISO 2000) de l'Organisation internationale de normalisation (ISO) énonce les critères d'une bonne gestion de la qualité système (SMQ). Généralement l'entreprise réalise un SQM afin de documenter les processus métier, la mise en place de mesure de contrôle et d'amélioration. Au travers de cette démarche l'entreprise documente également ces exigences en termes de ressource, personnel, formation, installation et environnement de travail.

En regardant l'architecture d'entreprise du point de vue de la gestion de la qualité en général et ISO 9001 en particulier, nous voyons sa contribution principale dans la conception intégrée, la gestion et la documentation de l'entreprise, de ses processus et de ses systèmes d'information.

Bien conçue et documentée, l'architecture d'entreprise permet à une organisation de se conformer aux exigences de la norme ISO 9001 relatives à l'identification des processus et de la documentation. De cette façon, la gestion de la qualité et de l'architecture d'entreprise forment une combinaison naturelle : La gestion de la qualité sera préoccupée par ce qui doit être conçu, documenté, contrôlé, mesuré, amélioré, … tandis que l'architecture d'entreprise déterminera la façon dont les processus et les ressources sont organisés et réalisés.

Gouvernance IT → COBIT

Selon COBIT (Control Objectives for Information and related Technology), une architecture d'entreprise bien définie est la base pour un bon environnement de contrôle interne. Dans de nombreuses entreprises, l'organisation informatique est responsable de l'établissement et du maintien de l'architecture d'entreprise. Considérant que COBIT se concentre sur la façon dont il faut organiser les fonctions de l'IT, l'architecture d'entreprise se concentre sur les structures métiers et informatiques, les processus, l'information et la technologie de l'entreprise. Ainsi, l'architecture d'entreprise constitue un complément naturel à COBIT.

Livraison et support IT → ITIL

ITIL (bibliothèque d'infrastructure IT) comprend une série de documents qui donnent des indications sur la mise à disposition des services informatiques ainsi que sur les installations nécessaires pour soutenir l'IT. L'approche d'ITIL est une approche de la gestion des services axée sur les processus. Il fournit des recommandations et des bonnes pratiques afin d'aider les entreprises à établir la gestion de la qualité de leur services IT ainsi que de leur infrastructure. Ici la «qualité» est définie comme «adaptés aux entreprises les besoins et les exigences des utilisateurs que ceux-ci évoluent. »

(Voir iso 20000)

La gestion des actifs informatiques d'une organisation est donc au cœur de l'ITIL. C'est à ce niveau qu'une architecture d'entreprise bien développée est très précieuse. Elle offre aux responsables informatiques une vision et une compréhension claire des applications informatiques, de l'infrastructure et des processus opérationnels liés, ainsi que des différentes dépendances entre ces deux domaines. Presque tous les processus de base identifiés par ITIL bénéficieront de l'architecture d'entreprise.

Implémentation IT → CMM/CMMI

CMMI (Capability Maturity Model Integration) a pour finalité essentielle de mesurer la capacité des projets à s'achever correctement en termes de délais, de fonctionnalités et de budget. Ce dernier est fondé à partir des bonnes pratiques de la profession, CMMI est structuré en 25 processus regroupés en 4 domaines :

- Process Management
- Project Management
- Engineering
- Support

CMMI définit aussi 5 niveaux de maturité et propose un modèle permettant de juger de la maturité d'une entreprise et de son système d'information.

Dans tout système d'information de taille importante, l'architecture logicielle joue un rôle important. La réalisation de cette architecture logicielle est encadrée par l'architecture d'entreprise qui fournit des contraintes et des lignes directrices aux projets de développement logiciel. En tant que tel, l'architecture d'entreprise est perçue comme quelque chose qui devient particulièrement utile (voire nécessaire) au CMMI Niveau 3 et au-delà. En effet, afin d'atteindre un tel niveau de maturité, il est nécessaire que tous les éléments logiciels de l'entreprise se conforment à des lignes directrices clairement définies.

A la lecture de ces quelques exemples, on peut constater que l'architecture d'entreprise constitue pour la plupart des cas présentés un complément ou une association naturelle. De manière plus générale, on peut relever que la capacité de connectivité d'un cadre d'architecture d'entreprise est un facteur important qui doit être pris en compte afin de juger de son applicabilité au sein d'une entreprise. En effet, le besoin de coexister avec d'autres cadres de management et de pilotage est souvent nécessaire afin de couvrir les domaines élicités dans le cadre d'un projet d'architecture d'entreprise.

Annexe 3 – Outils d'architecture d'entreprise

Le tableau suivant répertorie certains outils d'architecture d'entreprise répertoriés par Gartner et Forrester Research dans leurs rapports 2013 et 2014.[21] [22] [23]

Produit	Distributeur	Localisation	Dernière distribution stable	Date dernière distribution stable
ABACUS	Avolution	Australie	4.5	12/ 2015
ADOit	BOC Group	Autriche	7.0	06/2016
BiZZdesign Architect	BiZZdesign	Pays-Bas	4.8.0	12/2015
ARIS	Software AG	Allemagne	10.0	04/2017
Casewise Suite/Evolve	Casewise	Angleterre	2015/3.0	11/2015
Enterprise Architect	Sparx Systems	Australie	12	01/2015
iteraplan	iteratec	Allemagne	5.1	10/2015
leanIX	LeanIX	Allemagne	3.4	02/2016
Mega Suite	MEGA International	France	Release 3	08/2015
planningIT	Software AG	Allemagne	8.0	11/2012
SAP PowerDesigner	SAP-Sybase	Allemagne	16.1	11/2016
ProVision	OpenText	Canada	9.0	09/2012
QualiWare	QualiWare	Danemark	6.5	02/2017
SAMU	Atoll Technologies	Hongrie	5.6	12/2016
QPR EnterpriseArchitect	QPR Software	Finlande	2015.1	11/2015
System Architect	Unicomm	USA	11.4.3.6	12/2015
Troux	Troux Technologies	USA	9.1.2	03/2013

Figure 3-A : Liste d'outils d'architecture d'entreprise[24]

[21] Gartner Magic Quadrant for Enterprise Architecture Tools, 2013
[22] Forrester Wave EA Management Suites, Q2 2013
[23] Gartner Magic Quadrant for Enterprise Architecture Tools, 2014
[24] https://en.wikipedia.org/wiki/Enterprise_architecture

Annexe 4 – Vue d'ensemble de DDD (détaillée)

Cette figure présente l'ensemble des recommandations et techniques introduites par l'approche DDD.

Figure 4-A : DDD – Vue d'ensemble [DDD, 2004]

Annexe 5 – Définitions des éléments de base du métamodèle TOGAF

Le tableau suivant apporte une description des éléments de base du métamodèle TOGAF, ce tableau est issu de TOGAF9.1, section 34.5 Content Metamodel Entities :

Dénomination	Description
Actor	Un **acteur** est une personne, une organisation ou un processus déclenchant ou interagissant avec une ou plusieurs activités. Les acteurs peuvent être internes ou externes à une organisation. Par exemple, dans l'industrie automobile, un constructeur d'équipement d'origine serait considéré comme un acteur externe par un concessionnaire automobile qui interagit avec lui dans le cadre de ses activités d'approvisionnement.
Application Component	Un **composant applicatif** représente l'encapsulation d'une fonctionnalité applicative alignée sur la structure de mise en œuvre. Par exemple, le traitement d'une demande d'achat.
Assumption	Une **assertion** est une déclaration de fait probable qui n'a pas été entièrement validée à ce stade, en raison de contraintes externes. Par exemple, on peut supposer qu'une application existante prend en charge un ensemble d'exigences fonctionnelles, bien que ces exigences ne soient pas encore validées individuellement.
Business Service	Le **service métier** supporte les capacités métiers grâce à une interface définie et est explicitement géré par une organisation.
Capability	Une **capacité** offerte par une organisation, une personne ou un système. Les capacités sont exprimées en termes généraux, en utilisant des concepts d'un haut niveau d'abstraction. Elle nécessite pour sa mise en œuvre une combinaison d'organisations, de personnes, de processus et de technologies.
Constraint	Une **contrainte** est un facteur extérieur qui oblige une organisation à changer ses fonctionnements pour atteindre ses objectifs. Par exemple, les données des clients ne sont pas harmonisées au sein de l'organisation, à l'échelle régionale ou nationale, ce qui limite la capacité de l'organisation à offrir un service client efficace.
Data Entity	Une **entité de données** est une encapsulation de données reconnues par un expert du domaine d'entreprise comme une chose. Les entités de données logiques peuvent être liées aux applications, aux référentiels et aux services et peuvent être structurées en fonction des considérations d'implémentation.
Function	Une **fonction** est une fonctionnalité métier étroitement alignée à une organisation, mais pas nécessairement explicitement régies par l'organisation. Également appelée «fonction métier ».
Gap	Un **écart** est une déclaration de différence entre deux états. Utilisé dans le contexte de l'analyse des écarts, où la différence entre l'architecture de base et l'architecture cible est identifiée.
Organization Unit	Une **unité d'organisation** regroupe un ensemble de ressources et disposes de buts, objectifs et mesures. Les unités d'organisation peuvent inclure des parties externes et des organisations métier partenaires.
Platform Service	Un **service de plateforme** représente une capacité technique nécessaire pour fournir une infrastructure permettant de soutenir la livraison des applications.

Dénomination	Description
(Architecture) Principle	Un **principe d'architecture** est une déclaration qualitative qui devra être respectée ou mise en œuvre par l'architecture proposée, accompagnée de la ou des raisons nécessaires à sa mise en œuvre et de l'importance associée à cette mise en œuvre. Un principe d'architecture peut aussi être accompagné des raisons expliquant dans quels cas s'en passer. Remarque : Un exemple d'ensemble de principes d'architecture est défini dans la Partie III, 23. Principes d'architecture du document TOGAF.
Process	Un **processus** est une séquence d'activités qui ensemble atteignent un résultat spécifié, qui peut être décomposée en sous-processus, et qui montre l'opération d'une fonction ou d'un service (suivant le niveau de détail). Les processus peuvent aussi être utilisés pour relier ou composer des organisations, fonctions, services et processus.
Requirement	Une **exigence** est énoncé quantitatif des besoins métiers qui doit être satisfait par une architecture particulière ou un lot de travail.
Role	Un **rôle** est une fonction habituelle ou attendue d'un acteur, ou la part que quelqu'un ou quelque chose joue dans une action ou un événement spécifique. Un acteur peut avoir un certain nombre de rôles.
Technology Component	Un **composant technologique** représente l'encapsulation d'une infrastructure technique indépendante d'un produit particulier. Une classe de produit technique. Par exemple, un logiciel de logistique de la suite d'un progiciel de gestion intégré (ERP) ou un progiciel d'achat « sur étagère » (COTS).
Work Package	Un **paquet de travail** est un ensemble de tâches identifiées afin d'achever un ou plusieurs objectifs métiers. Un paquet de travail peut faire partie d'un projet, d'un projet complet ou d'un programme.

[25, 26]

[25] http://pubs.opengroup.org/architecture/togaf9-doc/arch/

[26] Open Group, *TOGAF 9.1 Translation Glossary : English – French (Document Number: C127)*, Open Group, 2012

Annexe 6 – Définitions des éléments d'extension du métamodèle TOGAF

Le tableau suivant apporte une description des éléments de base des extensions du métamodèle TOGAF, ce tableau est issu de TOGAF 9.1, section 34.5 Content Metamodel Entities :

Dénomination	Description
Contract	Un accord entre un consommateur de service et un fournisseur de service qui établit les paramètres fonctionnels et non fonctionnels de l'interaction.
Control	Une étape de prise de décision, accompagnée de la logique nécessaire à celle-ci, utilisée lors de l'exécution d'un processus, ou pour garantir que le processus soit conforme aux critères de gouvernance. Exemple: le contrôle de signature dans le processus de traitement de demandes d'achat.
Driver	Condition interne ou externe qui incite une organisation à définir ses objectifs. Un exemple de moteur externe serait un changement de réglementation ou de conformité aux règles qui exigerait un changement de la façon dont l'organisation fonctionne.
Event	Un changement de l'état d'une organisation qui déclenche un traitement. Les événements peuvent être générés à l'intérieur ou à l'extérieur de l'organisation et peuvent être résolus à l'intérieur ou à l'extérieur de l'organisation.
Goal	Une déclaration d'intention ou de direction de haut niveau d'une organisation. Généralement utilisé pour mesurer le succès d'une organisation.
Information System Service	Aspects automatisés d'un service métier. Un système d'information peut fournir ou supporter une partie ou la totalité d'un ou plusieurs services métiers.
Location	Un endroit où l'activité métier se déroule et peut être décomposé de façon hiérarchique.
Logical Data Component	Une zone délimitant les données encapsulées, relatives à une entité, de leur localisation physique où elles sont maintenues.
Logical Technology Component	Une encapsulation d'une infrastructure technique indépendante d'un produit particulier. Une classe de produit technique. Par exemple, un logiciel de logistique de la suite d'un progiciel de gestion intégré (ERP) ou un progiciel d'achat « sur étagère » (COTS).
Measure	Un indicateur ou un facteur qui peut être suivi, habituellement sur une base régulière, pour déterminer le succès ou l'alignement avec les objectifs et les buts.
Objective	Une étape temporellement bornée dans l'activité d'une organisation utilisée pour démontrer les progrès réalisés vers un objectif: par exemple, ''accroître l'utilisation des capacités de 30% d'ici la fin de 2009 pour soutenir l'augmentation prévue des parts de marché''.
Physical Application Component	Une application, un module applicatif, un service applicatif ou tout autre composant déployable d'une fonctionnalité. Par exemple, une instance configurée et déployée d'une application de gestion de la logistique de la Planification des ressources d'entreprise (ERP) commercialement disponible (COTS).
Physical Data Component	Une zone délimitée qui encapsule les entités liées entre elles de données pour former un emplacement mémoire. Par exemple, un ordre d'achat objet business, comprenant un entête et des noeuds d'achat.
Physical Technology Component	Un produit d'infrastructure technique spécifique ou une instance d'un produit d'infrastructure technique. Par exemple, une version particulière d'un produit commercialement disponible (COTS), ou une marque et une version spécifique d'un serveur.

Dénomination	Description
Product	Résultat généré par le métier. Le produit métier « commercial » de l'exécution d'un processus.
Requirement	Déclaration des besoins de l'entreprise qui doivent être comblés par une architecture particulière ou lot de travaux.
Service	Un élément de comportement qui fournit une fonctionnalité spécifique en réponse aux demandes d'acteurs ou d'autres services. Un service fournit ou contribue aux capacités de l'entreprise, à une interface explicitement définie et est explicitement régi. Les services sont définis pour le métier, le système d'information et les plates-formes.
Service Quality	Une configuration prédéfinie d'attributs non fonctionnels qui peut être assigné à un service ou un contrat de service.

[27, 28]

[27] http://pubs.opengroup.org/architecture/togaf9-doc/arch/

[28] Open Group, *TOGAF 9.1 Translation Glossary : English – French (Document Number: C127)*, Open Group, 2012

Annexe 7 – Contexte métier de mise en œuvre

Le contenu de cette annexe permet de découvrir les contextes informatique et organisationnel dans le cadre desquels l'utilisation conjointe de TOGAF et DDD est envisagée.

Présentation du contexte

Une bonne compréhension de l'environnement au sein duquel l'approche de conception guidée par le domaine est associée à la démarche d'architecture d'entreprise permettra de comprendre et de mieux cerner les objectifs de la démarche.

A cette fin, une présentation de la Direction de l'Octroi des aides agricoles (DO) est réalisée. Les défis majeurs relevés par le DO pour les années à venir et dont la mise en œuvre concerne en tout ou en partie son système d'information seront présentés. Pour ce faire nous nous appuierons sur une rapide description des situations initiales et cibles de son système d'information.

Contexte organisationnel

La Direction générale de l'Agriculture, des Ressources naturelles et de l'Environnement (DGO3) a été créée en 1993 au sein du Service Public de Wallonie (SPW), après la régionalisation des compétences agricoles, sur base de la loi spéciale du 8 août 1980, modifiée par les lois du 8 août 1988 et des 15 mai et 16 juillet 1993. La dernière phase de réforme de l'Etat, concrétisée par la loi du 13 juillet 2001 a donné aux Régions une compétence générale dans le domaine de la politique agricole, sans préjudice des compétences qui, à titre d'exception, relèvent toujours de l'Etat fédéral (essentiellement la sécurité de la chaîne alimentaire, dont notamment l'important secteur de la santé animale).

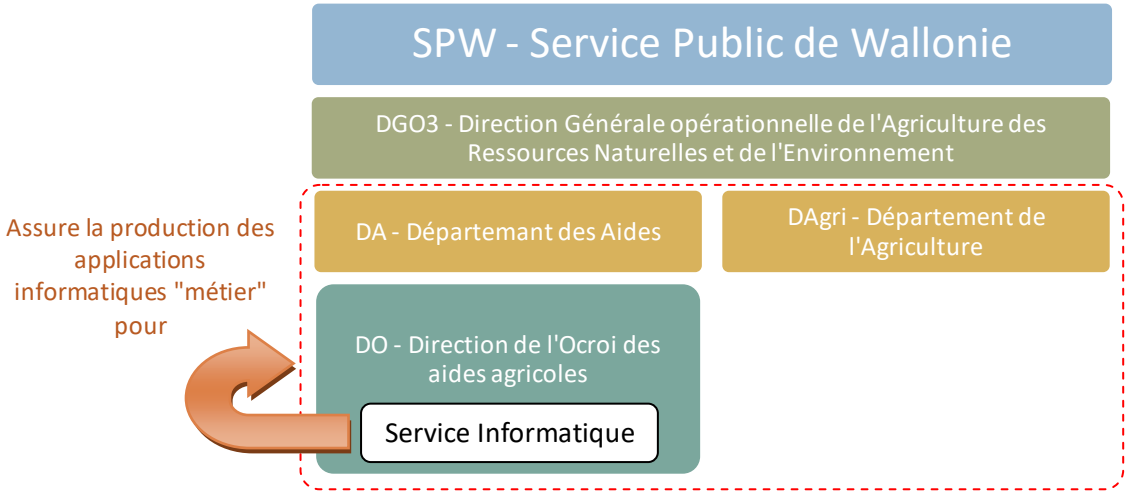

Figure 7-A - Organigramme SPW – DO [29]

Désormais, la DGO3 exerce, sur le territoire de la Wallonie, l'ensemble des compétences attribuées aux régions, c.-à-d. :

- aménagement du territoire
- rénovation rurale
- conservation de la nature
- politique agricole
- pêche maritime

Parmi les sous-structures composantes de la DGO3 (voir
Figure 7-A - Organigramme SPW – DO) se trouve plus spécifiquement le Département des Aides (DA) au sein duquel la Direction de l'Octroi des aides agricoles (DO) couvre certaines missions dont voici les principales :

[29] Service Public de Wallonie, *Présentation générale de la DGO3*, cellule de communication du SPW, 2015

- Assure le support à la gestion du département des Aides et du département de l'Agriculture en regard des obligations spécifiques à l'Organisme payeur de Wallonie (OPW) et en collaboration avec la Direction fonctionnelle et d'appui (DFA).

- Assure la production des applications informatiques "métier" en coordination avec le Département de l'Etude du Milieu Naturel et Agricole (DEMNA).

- Assure la disponibilité des infrastructures spécifiques et de l'informatique métier du système intégré de gestion et de contrôles (SIGEC) en coordination avec le DEMNA.

C'est dans le cadre de sa mission de « Production des applications informatiques métier » que la DO met en œuvre en son sein une démarche d'architecture d'entreprise et dont les chantiers de production informatique sont réalisés en respectant les paradigmes propres à l'approche de conception logicielle dirigée par le domaine.

Situation initiale du système d'information

La situation initiale du système d'information produit et maintenu par la Direction de l'Octroi des aides agricoles est un parc applicatif d'une centaine d'applications. Ce Parc d'application contient des applications dont les dates de mise en production et les durées de vie varient sensiblement, s'étalant de la période fédérale (1993) jusqu'à la période actuelle (2017).

Le patrimoine applicatif a été construit au fil du temps pour répondre aux différents besoins du Département des aides et de la Direction de l'agriculture. Il résulte d'une demande des métiers pour développer l'activité (faciliter par exemple la mise en place d'un nouveau régime d'aide), d'exigences opérationnelles (comme l'automatisation des processus métier), des contraintes techniques ou d'obligations réglementaires (liées à la PAC) et enfin, de décisions stratégiques prises par le Département des aides (développement, acquisitions de nouvelles compétences).

Par conséquent, l'accumulation de demandes successives des métiers, associée à une gouvernance parfois trop faible ainsi qu'à l'absence d'objectifs clairement définis, a généré un environnement applicatif complexe, hétéroclite, comportant des architectures techniques parfois très différentes, client lourd ou léger, langages allant du PowerBuilder au Java en passant par .NET, outils et librairies maison non standard, etc. (voir Figure 7-B - Répartition du parc applicatif DO par technologie). Lorsqu'elle existe, la documentation fonctionnelle et technique est de qualité variable. Le manque de vision transversale (vision en silo) a donné lieu à des redondances au niveau des données et de certaines fonctions métier ou de support.

Le poids du patrimoine applicatif se révèle par un chiffre simple : le coût de la maintenance de cet existant représente aujourd'hui une grande partie du budget IT et ne cesse de croître. Cela constitue une immobilisation des moyens humains et financiers considérable, qui grève l'investissement. On comprend mieux les enjeux colossaux de la gouvernance du Parc applicatif et de sa modernisation.

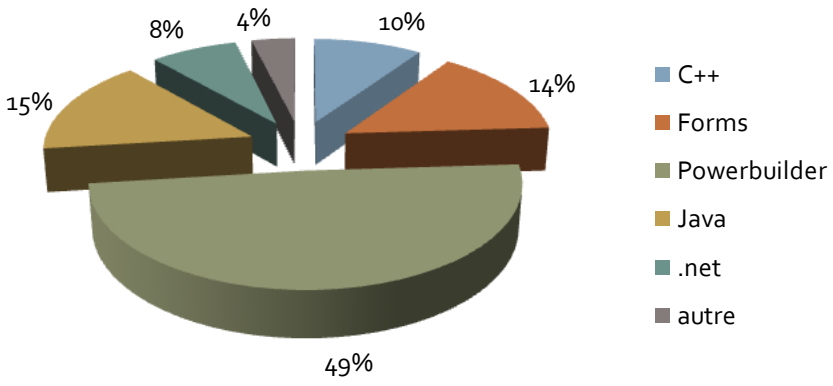

Figure 7-B - Répartition du parc applicatif DO par technologie [30]

[30] Service Public de Wallonie, *Présentation générale de la DGO3*, cellule de communication du SPW, 2015

Situation cible du système d'information

La situation cible souhaitée par la Direction de l'Octroi des aides agricoles comporte énormément de changements tant du point de vue de son métier que du point de vue de son système d'information.

Du point de vue métier la DO souhaite redéfinir sa relation avec les citoyens en s'inscrivant dans une démarche de dématérialisation et de simplification de ses services. En voici les points principaux :

- Création d'un Guichet WEB unique (accès aux services citoyens du DA)
- Harmonisation des procédures de traitement des différentes demandes d'aide agricoles
- Valorisation des données du DA (projet EWBS, source authentique)
- Simplification des processus métier
- Adoption des « standards » DTIC

D'un point de vue informatique la DO souhaite procéder à une refonte majeure de l'organisation interne de son département informatique et de son parc informatique. Cette refonte vise à atteindre les objectifs suivants :

- Amélioration des méthodologies de développement du SI
- Renforcement de la cohérence entre la gouvernance métier - IT
- Décloisonnement des applications
- Harmonisation de la vision d'architecture
- Harmonisation des technologies utilisées au sein du SI
- Mise à jour technologique
- Diminution des coûts de maintenance et d'infrastructure
- Renforcement de l'implication du métier dans le cycle de vie logiciel

Indubitablement, pour atteindre un tel objectif et compte tenu de la situation initiale du parc informatique de la DO, il semble évident qu'une majorité des applications du parc doivent être remplacées. Bien que l'obsolescence des technologies employées semble la plupart du temps justifier à elle seule ce choix, le cloisonnement fonctionnel des applications oblige lui aussi à un remplacement de la plupart de ces applications.

Afin de se structurer et de s'organiser face aux changements en cours et à venir, la DO souhaite procéder à la création d'un cadre de gouvernance global permettant d'optimiser la gouvernance IT-métier durant et après cette période de changement important. Pour ce faire la DO met en place une démarche d'architecture d'entreprise en son sein. Cette démarche doit permettre d'arriver à une gouvernance plus efficace. En plus de documenter la situation initiale et la situation cible elle permettra d'établir les étapes nécessaires à l'évolution du parc informatique. Durant et après cette période de transition elle facilitera la compréhension entre le métier et l'IT. Une meilleure communication entre le métier et l'IT est une condition sinequanone pour permettre l'alignement du plan IT sur le plan métier et ainsi garantir que les financements IT futurs seront consacrés en priorité aux projets les plus générateurs de valeur.

Parallèlement à la démarche d'architecture d'entreprise, le DO souhaite se doter d'une vision commune concernant le style d'architecture utilisé dans le cadre de sa conception logicielle. Une harmonisation de la vision d'architecture applicative doit permettre de diminuer les coûts liés aux nombreuses expertises requises actuellement. De plus, cette harmonisation facilitera l'interopérabilité des personnes au sein des différents projets, ce qui contribuera directement à l'amélioration de l'agilité du service informatique.

D'autres actions sont mises en œuvre actuellement pour soutenir cette évolution importante de la DO. Parmi celles-ci, citons encore :

- Réorganisation du service informatique
    - Redistribution et raffinement des responsabilités
    - Création d'une cellule PMO (Project Management Office)
    - Création d'une cellule d'Urbanisation (réalise la démarche d'AE)
    - Création d'une notion de « Mentor » (référant)
- Modification importante de l'infrastructure technique et de sa gestion
- Adoption d'outils standards et constitution d'un unique cadre conceptuel de développement

Parmi les actions entreprises au sein de la DO pour soutenir ses efforts, deux éléments suscitent notre intérêt dans le cadre de ce mémoire : la démarche d'architecture d'entreprise et l'harmonisation de la vision d'architecture applicative. La réussite de mise en œuvre conjointe de ces éléments est un levier jouant un rôle important dans la transformation souhaitée par la DO.

Le choix réalisé par la DO concernant la méthodologie de définition du cadre de gouvernance IT-métier s'est naturellement porté sur TOGAF, qui d'une part propose une démarche soutenant l'objectif de gouvernance que s'est fixée la DO, et d'autre part est un cadre de définition d'architecture d'entreprise faisant partie des standards recommandés par le DTIC et sur-lesquels le SPW possède une expertise non-négligeable.

En ce qui concerne le choix retenu dans le cadre de mise en place d'une vision d'architecture commune pour la conception de ses futures applications, la DO à souhaité sélectionner une approche valorisant son capital principal : son domaine métier. Sur base de cette exigence, des différentes opportunités qui se présentaient et des expertises dont elle dispose, la DO à pris la décision d'adopter une méthode de conception logicielle dirigée par le domaine (Domain-Driven Design - DDD).